**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»**

**ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №4**

Теория информации

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

Руководитель А. Н. Бочаров

подпись, дата инициалы, фамилия

Обучающийся БПИ22-02, 221219040 К. В. Трифонов

номер группы, зачетной книжки подпись, дата инициалы, фамилия

Красноярск 2024 г.

# ЦЕЛЬ РАБОТЫ

Закрепление знаний по методам кодирования информации.

# задания

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

Код должен предусматривать возможность посылки *n* сообщений.

1. Рассчитать параметры кода: *n*и, *n*к, *n*.
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 2**:

кол-во сообщений *N* = 32.

# ХОД РАБОТЫ

**Задание 1.**

Количество сообщений: *N* = 32.

Разрядность информационной части кода: *n*и = log2 *N* = log2 32 = 5 бит.

Разрядность контрольной части кода: *n*к = log2 ((*n*и + 1) + log2 (*n*и + 1)) =

= log2 ((5 + 1) + log2 (5 + 1)) ≈ 3,1 = 4 бита.

Общая разрядность кода: *n* = *n*и + *n*к = 5 + 4 = 9 бит.

**Задание 2.**

Пример построения расширенного кода Хемминга:

Случайное кодовое сообщение: 10010

*n*и = 5 бит, *n*к = 4 бита, *n* = 9 бит.

**Заполним контрольные заряды 1, 2, 4, 8:**

1 проверка: a1 + a3 + a5 + a7 + a9 = к1 + 1 + 0 + 1 + 0 = 0 – чётное кол-во единиц => к1 = 0.

2 проверка: a2 + a3 + a6 + a7 = к2 + 1 + 0 + 1 = 0 – чётное кол-во единиц => к2 = 0.

3 проверка: a4 + a5 + a6 + a7 = к3 + 0 + 0 + 1 = 0 – нечётное кол-во единиц => к3 = 1.

4 проверка: a8 + a9 = к4 + 0 = 0 – чётное кол-во единиц => к4 = 0.

Готовый код представлен в таблице 1.

Таблица 1 – Код Хемминга

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| a1 | a2 | a3 | a4 | a5 | a6 | a7 | a8 | a9 |
| 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 |

001100100к6 – кол-во единиц нечётное => к6 = 1

0011001001 – расширенный код Хемминга.

10 составленных кодовых сообщений приведены в таблице 2.

Таблица 2 – Коды Хемминга

|  |  |
| --- | --- |
| Кодовое сообщение | Код Хемминга |
| 00110 | 1000011001 |
| 01011 | 1100101110 |
| 01010 | 0100101001 |
| 10101 | 0011010111 |
| 01010 | 0100101001 |
| 10001 | 0110000110 |
| 01110 | 0001111000 |
| 00000 | 0000000000 |
| 01001 | 0001100110 |
| 00011 | 0101001111 |

**Задание 3.**

Процедура исправления ошибки в одной из позиций:

Пусть присутствует ошибка в 1 разряде (таблица 3).

Таблица 3 – Код с ошибкой

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **a1** | a2 | a3 | a4 | a5 | a6 | a7 | a8 | a9 | a10 |
| **0** | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 1 |
| **1** | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 1 |

1011001001 – нечётное кол-во единиц => единичная ошибка.

**Проверки:**

1 проверка: a1 + a3 + a5 + a7 + a9 = 1 + 1 + 0 + 1 + 0 = 1 => s1 = 1.

2 проверка: a2 + a3 + a6 + a7 = 0 + 1 + 0 + 1 = 0 => s2 = 0.

3 проверка: a4 + a5 + a6 + a7 = 1 + 0 + 0 + 1 = 0 => s3 = 0.

4 проверка: a8 + a9 = 0 + 0 = 0 => s4 = 0.

Вектор ошибки: s = 00001 = 110 => ошибка в 1 разряде.

**Процедура обнаружения двойной ошибки:**

Пусть присутствуют ошибки в 1 и 2 разрядах (таблица 4):

Таблица 4 – Код с ошибкой

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **a1** | **a2** | a3 | a4 | a5 | a6 | a7 | a8 | a9 | a10 |
| **0** | **0** | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 1 |
| **1** | **0** | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 1 |

1011001001 – чётное кол-во единиц , значит либо ошибки нет, либо ошибка двойная.

**Проверки:**

1 проверка: a1 + a3 + a5 + a7 + a9 = 1 + 1 + 0 + 1 + 0 = 1 => s1 = 1.

2 проверка: a2 + a3 + a6 + a7 = 0 + 1 + 0 + 1= 0 => s2 = 0.

3 проверка: a4 + a5 + a6 + a7 + = 1 + 0 + 0 + 1= 0 => s3 = 0.

4 проверка: a8 + a9 = 0 + 0 = 0 => s4 = 0.

Вектор ошибки s = 00001 не равен 0, значит ошибка присутствует в коде.

**Задание 4.**

Текст программы представлен на листинге 1.

Листинг 1 – Текст программы

import random

#Задание списка кодов

def gen(length=5, count=10):

    binary\_list = []

    for \_ in range(count):

        binary\_num = ''.join(str(random.randint(0, 1)) for \_ in range(length))

        binary\_list.append(binary\_num)

    return binary\_list

#Расширенный код Хемминга

def hamCode(n):

    k1 = ((n[0] + n[1] + n[3] + n[4]).count('1')) % 2

    k2 = ((n[0] + n[2] + n[3]).count('1')) % 2

    k3 = ((n[1] + n[2] + n[3]).count('1')) % 2

    k4 = ((n[4]).count('1')) % 2

    result = str(k1) + str(k2) + n[0] + str(k3) + n[1:4] + str(k4) + n[4:]

    if (result.count('1') % 2) == 0:

        hamming\_code = result + '0'

    else:

        hamming\_code = result + '1'

    return hamming\_code

#Одиночная ошибка

def single\_mistake(code):

    code\_list = list(code)

    index = random.randint(0, 9)

    if code\_list[index] == '1':

        code\_list[index] = '0'

    else:

        code\_list[index] = '1'

    Icode = ''.join(code\_list)

    return Icode, index+1

#Двойная ошибка

def doub(code):

    code\_list = list(code)

    indices = random.sample(range(10), 2)

    for index in indices:

        if code\_list[index] == '1':

            code\_list[index] = '0'

        else:

            code\_list[index] = '1'

    dIcode = ''.join(code\_list)

    return dIcode, (indices[0]+1, indices[1]+1)

#Обнаружение ошибок

def detector(code):

    s1 = ((code[0] + code[2] + code[4] + code[6] + code[8]).count('1')) % 2

    s2 = ((code[1] + code[2] + code[5] + code[6]).count('1')) % 2

    s3 = ((code[3] + code[4] + code[5] + code[6]).count('1')) % 2

    s4 = ((code[7] + code[8]).count('1')) % 2

    s = str(s4) + str(s3) + str(s2) + str(s1)

    if (code.count('1') % 2) == 1:

        return s

    else:

        if int(s, 2) == 0:

            return 'Ошибки нет'

        else:

            return s

#Исправление одиночной ошибки

def corrector(code, syndrome):

    s = int(syndrome, 2)

    code\_list = list(code)

    if code\_list[s-1] == '1':

        code\_list[s-1] = '0'

    else:

        code\_list[s-1] = '1'

    correct\_code = ''.join(code\_list)

    return correct\_code

if \_\_name\_\_ == "\_\_main\_\_":

    bina = gen()

    hamm = []

    for n in bina:

        hamm.append(hamCode(n))

    Imiss = []

    ImissID = []

    for code in hamm:

        Icode, single\_mistake\_index = single\_mistake(code)

        Imiss.append(Icode)

        ImissID.append(single\_mistake\_index)

    IImiss = []

    IImissID = []

    for code in hamm:

        dIcode, doub\_index = doub(code)

        IImiss.append(dIcode)

        IImissID.append(doub\_index)

    Isyn = []

    for code in Imiss:

        Isyn.append(detector(code))

    dIsyn = []

    for code in IImiss:

        dIsyn.append(detector(code))

    corr = []

    for i in range(len(Imiss)):

        corr.append(corrector(Imiss[i], Isyn[i]))

    print(f'{str("Код"):^16}| {str("Код Хемминга"):^16}')

    print("----------------|-----------------")

    for i in range(len(bina)):

        print(f'{bina[i]:^16}|{hamm[i]:^16}')

    print(" Код Хемминга   | 1 ошибка        | Ошибочный разряд    | "

        "    Симптом      | Исправленный код")

    print("----------------|-----------------|---------------------|------------------|----------")

    for i in range(len(hamm)):

        print(f'{hamm[i]:^15} | {Imiss[i]:^15} | {ImissID[i]:^20}'

            f'| s = {Isyn[i]} = 'f'{int(Isyn[i], 2)}     | {corr[i]}')

    print(f'{str("Двойная ошибка"):^100}')

    print(f'{str("Код Хемминга"):^16}| {str("2 ошибки"):^15} | {str("Ошибочные разряды"):^19} | {str("Симптом"):^16}')

    print("----------------|-----------------|---------------------|-----------------")

    for i in range(len(hamm)):

        print(f'{hamm[i]:^15} | {IImiss[i]:^15} | {(IImissID[i])[0]:^9} |'

            f'{(IImissID[i])[1]:^9}| s = {dIsyn[i]:^}')

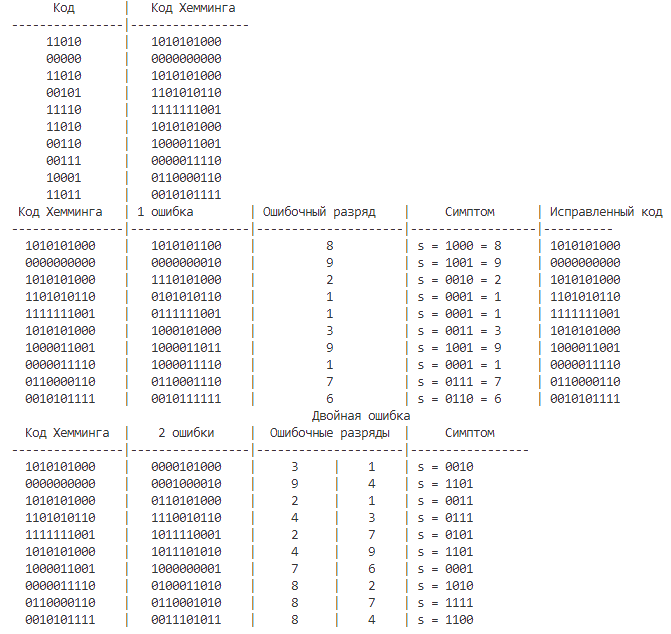
Результат работы программы изображен на рисунке 1. 

Рисунок 1 – работа программы

Программа генерирует случайные кодовые сообщения, состоящие из 5 разрядов, кодирует их кодом Хемминга и проводит тестирование с помощью обнаружения и исправления 1 ошибки, обнаружения 2 ошибок. Затем результаты работы выводятся в таблицы.

# контрольные вопросы

1. Что такое корректирующий код?

Корректирующий код — это специальный код, используемый для обнаружения и исправления ошибок при передаче данных через шумный канал.

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

Ошибки обнаруживаются с помощью проверки кодового слова на соответствие заданным правилам, таким как проверка наличия четного или нечетного количества единиц в коде.

3. Как рассчитать параметры корректирующего кода?

Параметры корректирующего кода, такие как длина кода, количество информационных и проверочных бит, выбираются на основе требуемого уровня исправления ошибок и надежности.

4. Как определить кодовое расстояние?

Кодовое расстояние — это минимальное количество изменений, необходимое для превращения одного кодового слова в другое. Оно может быть определено путем нахождения расстояния Хэмминга между всеми парами кодовых слов в коде.

5. Как определить минимальное кодовое расстояние?

Минимальное кодовое расстояние — это минимальное кодовое расстояние в коде. Оно представляет собой минимальное количество изменений, необходимое для различения любых двух кодовых слов.

6. Как строится код Хемминга.

Код Хэмминга строится путем размещения информационных битов в правильных позициях и вычисления контрольных битов таким образом, чтобы обнаруживать и исправлять ошибки.

7. Как проводится процедура декодирования кода Хемминга?

Для декодирования кода Хэмминга производится проверка контрольных битов на наличие ошибок, и при их обнаружении проводится процедура исправления ошибок с использованием информационных и контрольных битов.

8. Что такое пропускная способность канала связи?

Пропускная способность канала связи — это максимальное количество информации, которое может быть передано через канал за единицу времени.

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

Теорема Шеннона утверждает, что существует предельная пропускная способность для дискретного канала с помехами, которая может быть достигнута с помощью кодирования при условии, что ошибка в передаче стремится к нулю.

10. Как описать двоичный канал?

Двоичный канал — это канал связи, в котором передается информация в виде двоичных символов (обычно 0 и 1).

11. Как определить пропускную способность двоичного канала?

Пропускная способность двоичного канала определяется как максимальная скорость передачи информации, которая может быть достигнута при заданном уровне шума и ошибок в канале.

# ВЫВОДЫ

В ходе выполнения лабораторной работы мной были закреплены знания по методам кодирования информации.