Министерство образования и науки РФ

Федеральное государственное автономное

образовательное учреждение высшего образования

«Санкт-Петербургский национальный исследовательский университет

информационных технологий, механики и оптики»

**факультет программной инженерии и компьютерной техники**

**ЛАБОРАТОРНАЯ РАБОТА № 2**

по дисциплине

‘ИНФОРМАТИКА’

‘Синтез помехоустойчивого кода’

Вариант №58

*Выполнил:*

Студент группы P3118

Павлов Александр Сергеевич

*Преподаватель:*

Балакшин Павел Валерьевич



Санкт-Петербург, 2021

Оглавление

[Задания: 3](#_Toc83764975)

[Основные этапы вычисления: 3](#_Toc83764976)

[Вывод: 8](#_Toc83764977)

[Список литературы: 8](#_Toc83764978)

# Задания:

Проверить двоичный код на ошибочность, если есть ошибки – исправить:

№43: 0000011 (классический код Хэмминга (7,4))

№75: 0101101 (классический код Хэмминга (7,4))

№107: 1001111 (классический код Хэмминга (7,4))

№27: 1110001 (классический код Хэмминга (7,4))

№58: 010001110100011 (классический код Хэмминга (15,11))

Сложить номера всех 5 вариантов заданий, умножить полученное число на 4. Принять данное число как число информационных разрядов в передаваемом сообщении. Вычислить для данного числа минимальное число проверочных разрядов и коэффициент избыточности.

Написать программу на любом языке программирования, которая на вход из командной строки получает набор из 7 цифр «0» и «1», записанных подряд, анализирует это сообщение на основе классического кода Хэмминга (7,4), а затем выдает правильное сообщение (только информационные биты) и указывает бит с ошибкой при его наличии.

# Основные этапы вычисления:

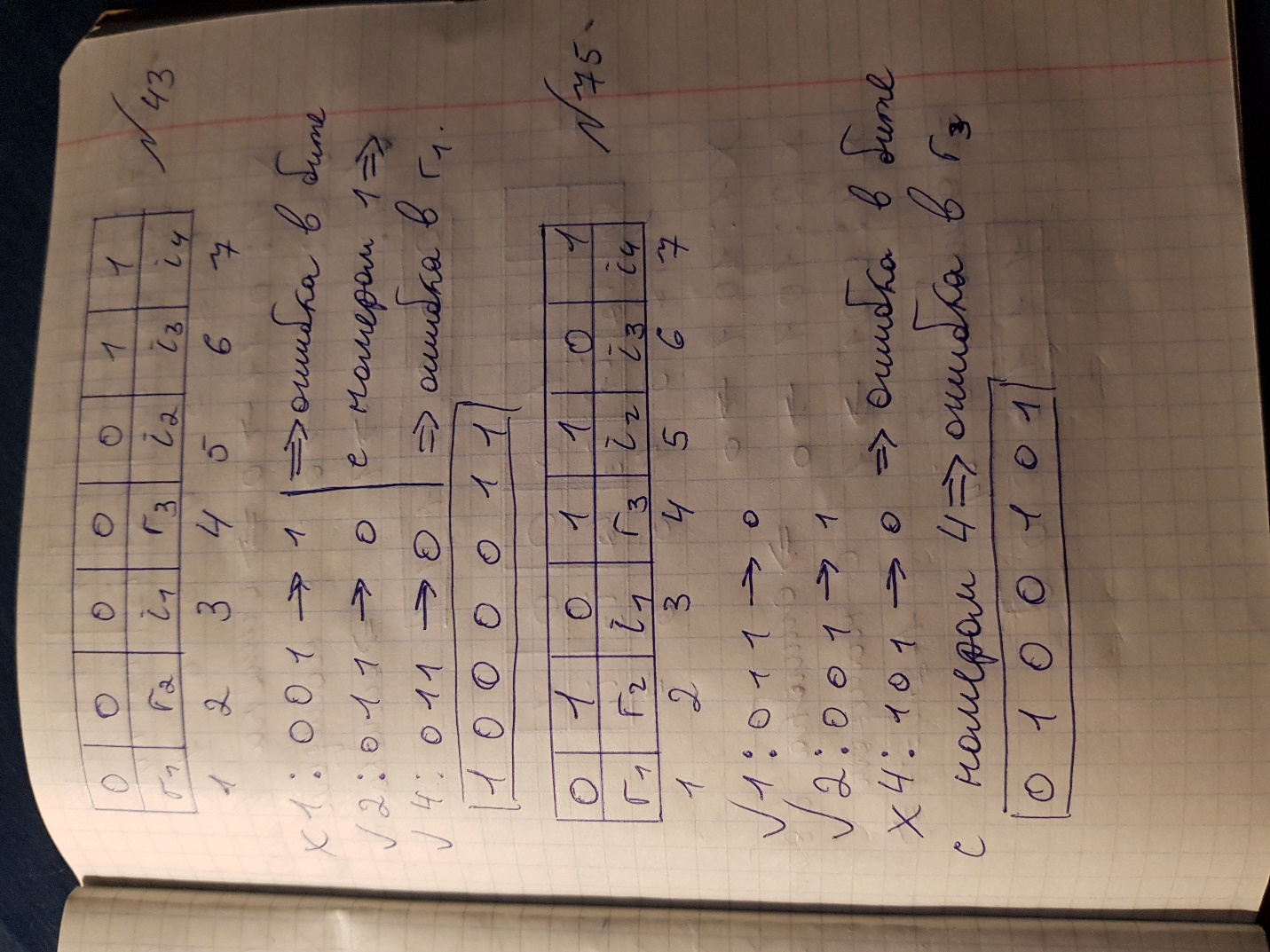


Рисунок 1 – схема декодирования кода Хэмминга (7,4)

Первый проверочный бит: i­1 ⊕ i2 ⊕i4  = 0 ⊕ 0 ⊕ 1 = 1

Второй проверочный бит: i­1 ⊕ i3 ⊕i4  = 0 ⊕ 1 ⊕ 1 = 0

Третий проверочный бит: i­2 ⊕ i3⊕i4  = 0 ⊕ 1 ⊕ 1 = 0

Так как первый проверочный бит показал ошибку, а остальные нет, то ошибка в бите с номером 1.

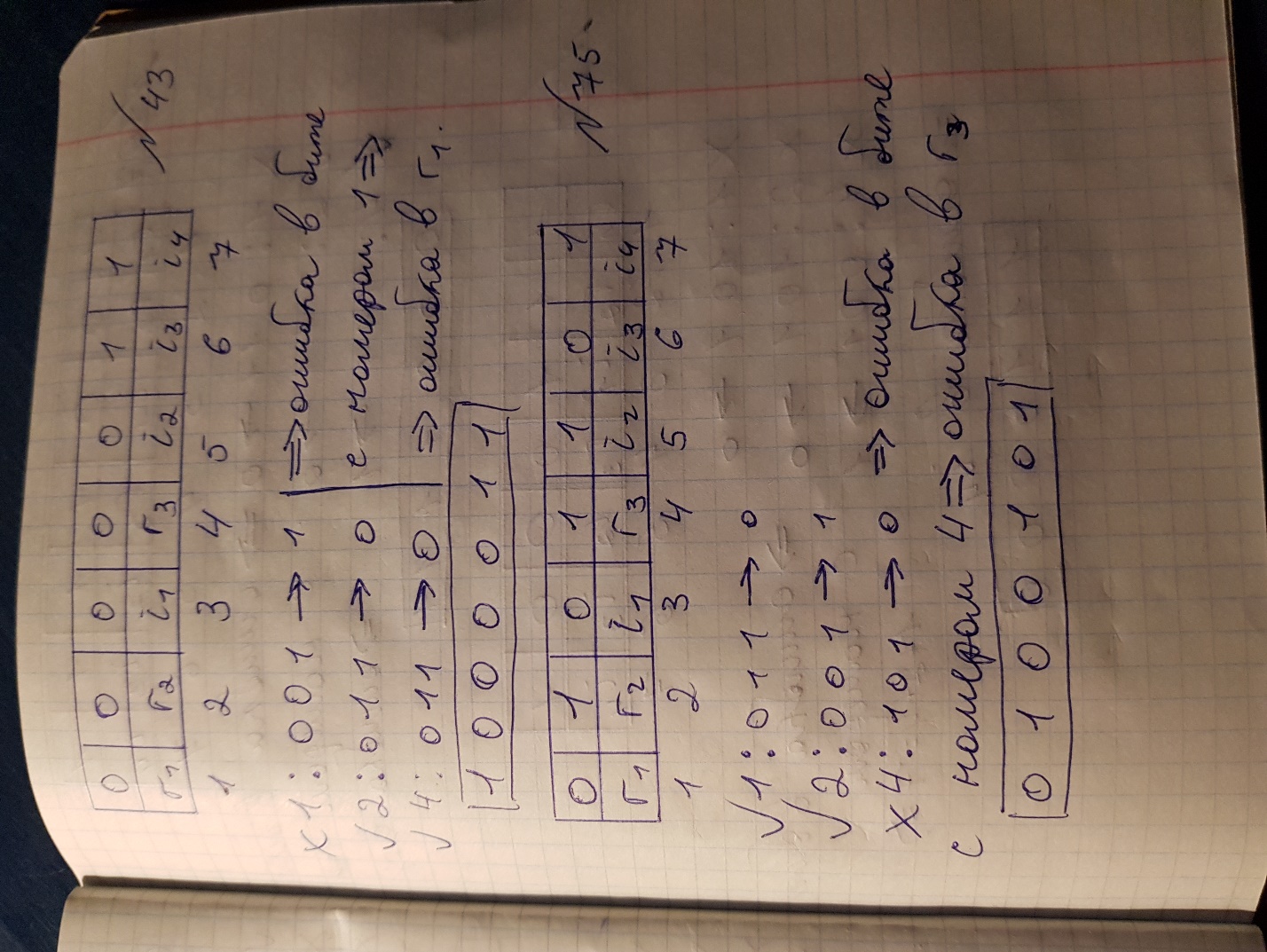


Рисунок 2 – схема декодирования кода Хэмминга (7,4)

Первый проверочный бит: i­1 ⊕ i2 ⊕i4  = 0 ⊕ 1 ⊕ 1 = 0

Второй проверочный бит: i­1 ⊕ i3 ⊕i4  = 0 ⊕ 0 ⊕ 1 = 1

Третий проверочный бит: i­2 ⊕ i3⊕i4  = 1 ⊕ 0 ⊕ 1 = 0

Так как третий проверочный бит показал ошибку, а остальные нет, то ошибка в бите с номером 4.

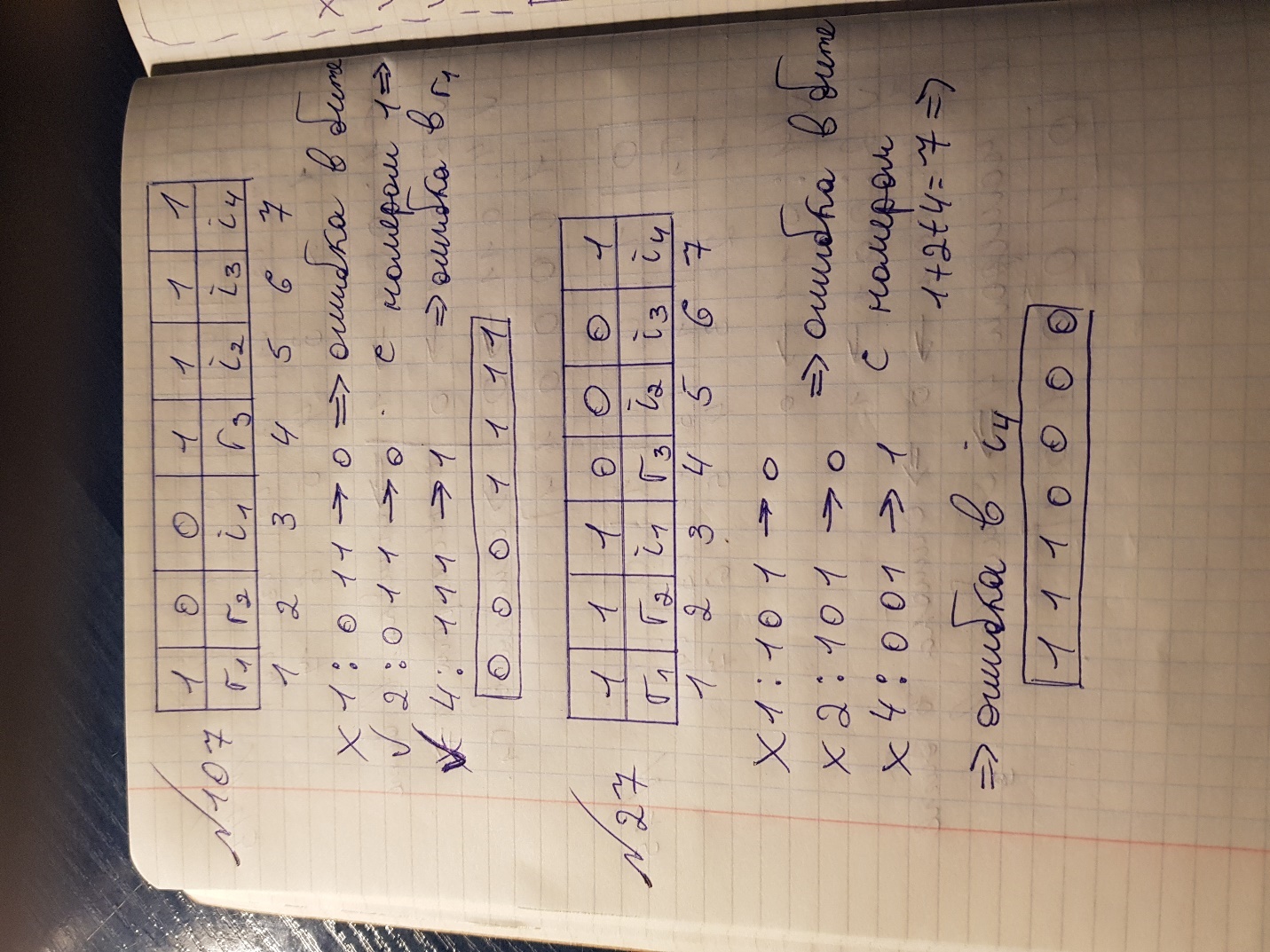


Рисунок 3 – схема декодирования кода Хэмминга (7,4)

Первый проверочный бит: i­1 ⊕ i2 ⊕i4  = 0 ⊕ 1 ⊕ 1 = 0

Второй проверочный бит: i­1 ⊕ i3 ⊕i4  = 0 ⊕ 1 ⊕ 1 = 0

Третий проверочный бит: i­2 ⊕ i3⊕i4  = 1 ⊕ 1 ⊕ 1 = 1

Так как первый проверочный бит показал ошибку, а остальные нет, то ошибка в бите с номером 1.

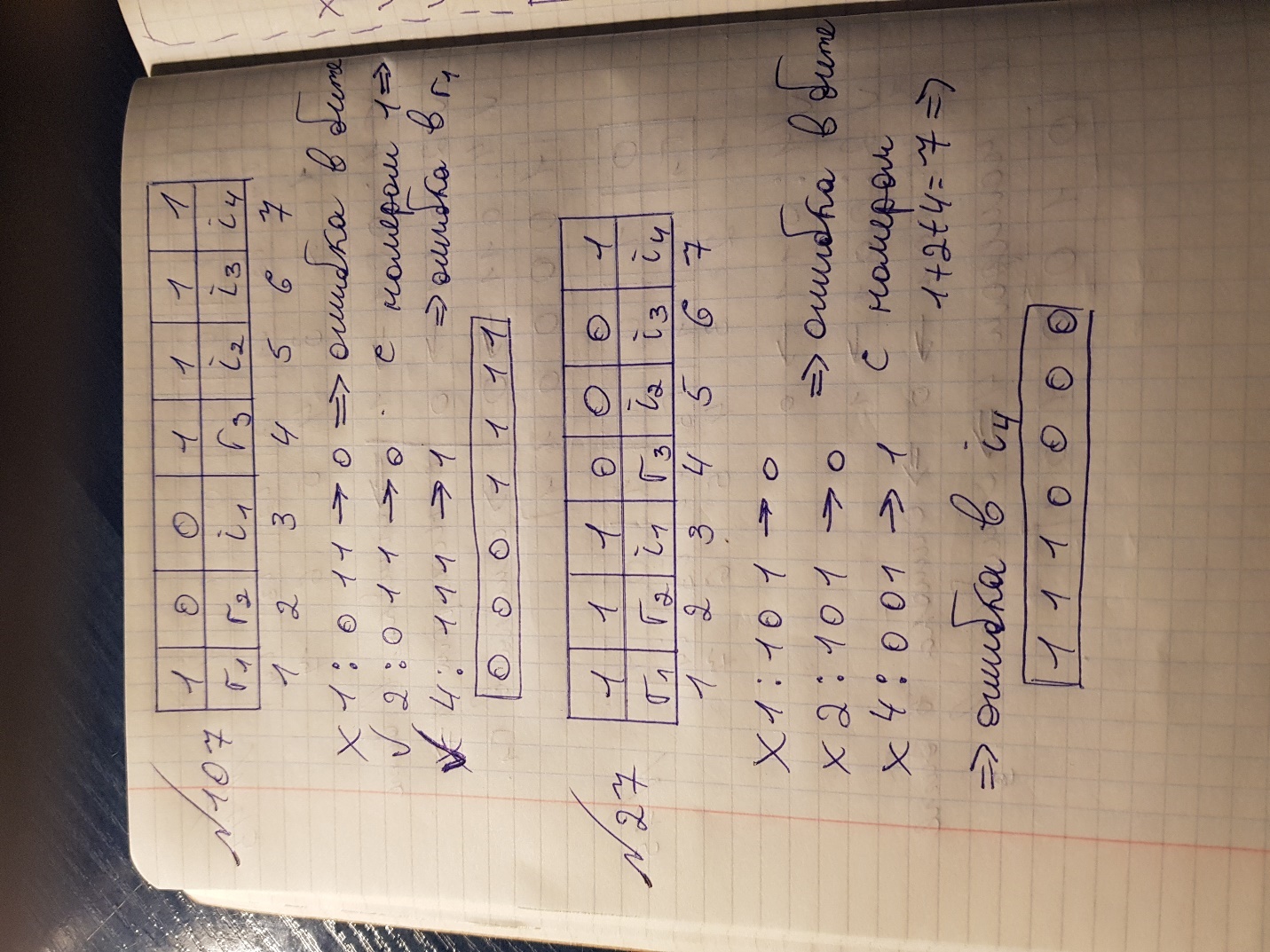


Рисунок 4 – схема декодирования кода Хэмминга (7,4)

Первый проверочный бит: i­1 ⊕ i2 ⊕i4  = 1 ⊕ 0 ⊕ 1 = 0

Второй проверочный бит: i­1 ⊕ i3 ⊕i4  = 1 ⊕ 0 ⊕ 1 = 0

Третий проверочный бит: i­2 ⊕ i3⊕i4  = 0 ⊕ 0 ⊕ 1 = 1

Так как первый, второй и третий проверочный биты показали ошибку, то ошибка в бите с номером 1 + 2 + 4 = 7.

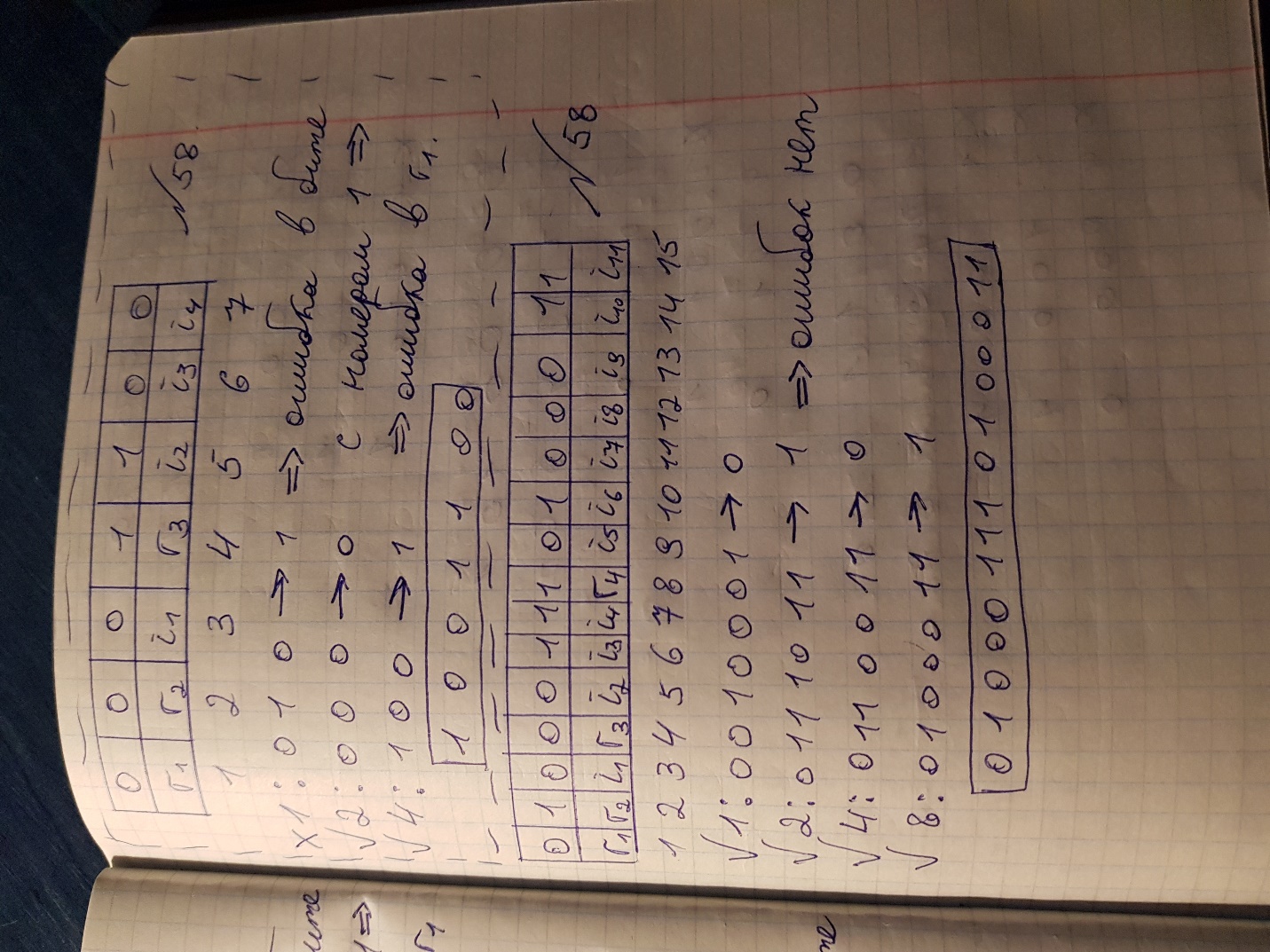


Рисунок 4 – схема декодирования кода Хэмминга (15,11)

Первый проверочный бит: i­1 ⊕ i2 ⊕i4  ⊕ i5 ⊕ i7 ⊕ i9 ⊕ i11= 0 ⊕ 0 ⊕ 1 ⊕ 0 ⊕ 0 ⊕ 0 ⊕ 1 = 0

Второй проверочный бит: i­1 ⊕ i3 ⊕i4 ⊕ i6 ⊕ i7 ⊕ i10 ⊕ i11 = 0 ⊕ 1 ⊕ 1 ⊕ 1 ⊕ 0 ⊕ 1 ⊕ 1 = 1

Третий проверочный бит: i­2 ⊕ i3⊕i4 ⊕ i8 ⊕ i9 ⊕ i10 ⊕ i11 = 0 ⊕ 1 ⊕ 1 ⊕ 0 ⊕ 0 ⊕ 1 ⊕ 1 = 0

Четвертый проверочный бит : i5 ⊕ i6 ⊕ i7 ⊕ i8 ⊕ i9 ⊕ i10 ⊕ i11 = 0 ⊕ 1 ⊕ 0 ⊕ 0 ⊕ 0 ⊕ 1 ⊕ 1 = 1

Так как ни первый, ни второй, ни третий, ни четвертый проверочные биты не оказались ошибочными, то код доставлен правильно(ошибок нет).

Сумма всех 5 вариантов заданий = 43 + 75 + 107 + 27 + 58 = 310

Сумма, умноженная на 4 = 310 \* 4 = 1240.

По формуле 2r ≥ r + i + 1, где r – минимальное кол-во проверочных битов, i – кол-во информационных битов получаем:

2r – r ≥ 1241 => r = 11 – минимальное кол-во проверочных битов для 1240 информационных битов.

Коэффициент избыточности вычисляется по формуле

Получаем коэффициент избыточности = ~ 0,0088.

Программа для анализа кода Хэмминга на языке Python:

a = input()

flag = 0

if len(a) != 7:

print('Длина набора не равна 7.')

flag = 1

else:

for i in a:

if int(i) != 0 and int(i) != 1:

print('Не двоичная запись числа')

flag = 1

if flag == 0:

k = 0

r1 = a[0]

r2 = a[1]

r3 = a[3]

modr1 = (a[2] + a[4] + a[6]).count('1') % 2

modr2 = (a[2] + a[5] + a[6]).count('1') % 2

modr3 = (a[4] + a[5] + a[6]).count('1') % 2

if r1 != str(modr1):

k += 1

if r2 != str(modr2):

k += 2

if r3 != str(modr3):

k += 4

if (k > 0):

if k == 1:

if a[k-1] == '0':

a = '1' + a[k:]

else:

a = '0' + a[k:]

elif k == 7:

if a[k-1] == '0':

a = a[:k-1] + '1'

else:

a = a[:k-1] + '0'

elif a[k-1] == '0':

a = a[:k-1] + '1' + a[k:]

else:

a = a[:k-1] + '0' + a[k:]

print(f'Ошибка в бите №{k}')

print('Правильное сообщение(только информационные биты) -',a[2]+a[4]+a[5]+a[6])

else:

print('Ошибок нет, код доставлен верно')

print(a[2]+a[4]+a[5]+a[6])

# Вывод:

В ходе выполнения данной лабораторной работы я познакомился с понятием помехоустойчивых кодов, в частности – с кодом Хэмминга. Научился декодировать и проверять на правильность такие коды. Алгоритм кодирования Хэмминга - очень популярен и позволяет значительно повысить надежность передачи и хранения информации.

# Список литературы:

1. Презентация «Код Хэмминга» Балакшин П.В – 2021 год [Электронный ресурс]. –URL:

<https://isu.ifmo.ru/pls/apex/f?p=2002:0:100380356337453:DWNLD_F:NO::FILE:240C9DC86073CEE69447B8F41E752787>

1. Питерсон У., Уэлдон Э. Коды, исправляющие ошибки: Пер. с англ. М.: Мир - 1976 год [Электронный ресурс]. – URL:

<https://www.studmed.ru/view/piterson-u-ueldon-e-kody-ispravlyayuschie-oshibki_9657dd030d4.html>