Федеральное государственное автономное образовательное учреждение высшего образования **«Национальный исследовательский университет ИТМО»**

Факультет Программной Инженерии и Компьютерной Техники

**Лабораторная работа по информатике №2**

Вариант: 75

Преподаватель: Рудникова Тамара Владимировна

Выполнил: Терехин Никита Денисович

Группа: Р3108

Санкт-Петербург

2022г

Оглавление

[Задание 1 3](#_Toc116433180)

[Ход выполнения задания 3](#_Toc116433181)

[Задание 2 7](#_Toc116433182)

[Задание 3 8](#_Toc116433183)

[Задание\* 8](#_Toc116433184)

[Исходный код на Java 9](#_Toc116433185)

[Результат работы программы 9](#_Toc116433186)

[Вывод 10](#_Toc116433187)

[Список литературы. 10](#_Toc116433188)

## Задание 1

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Вариант** | **1** | | | | **2** |
| 75 | 57 | 94 | 19 | 10 | 74 |

Входные варианты сообщений:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **ALT** | **1** | **2** | **3** | **4** | **5** | **6** | **7** |
| **r1** | **r2** | **i1** | **r3** | **i2** | **i3** | **i4** |
| **10** | 1 | 0 | 1 | 0 | 0 | 0 | 0 |
| **19** | 0 | 1 | 0 | 1 | 0 | 0 | 1 |
| **57** | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| **94** | 1 | 0 | 1 | 0 | 1 | 1 | 0 |

Построить схему декодирования классического кода Хэмминга (7;4), которую представить в отчёте в виде изображения. Показать, исходя из выбранных вариантов сообщений, имеются ли в принятом сообщении ошибки, и если имеются, то какие. **Подробно прокомментировать** и записать правильное сообщение.

## Ход выполнения задания

Таблица 1 – Таблица кодирования информационных битов

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **1** | **2** | **3** | **4** | **5** | **6** | **7** |  |
| **2n** | *r1* | *r2* | *i1* | *r3* | *i2* | *i3* | *i4* | S |
| **1** | x |  | x |  | x |  | x | *s1* |
| **2** |  | x | x |  |  | x | x | *s2* |
| **4** |  |  |  | x | x | x | x | *s3* |

**Для первого примера имеем:**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **r1** | **r2** | **i1** | **r3** | **i2** | **i3** | **i4** |
| 1 | 0 | 1 | 0 | 0 | 0 | 0 |

Посчитаем контрольную сумму для каждого проверочного разряда. Для этого нужно произвести суммирование по модулю 2 контрольного бита и информационных битов, которые им кодируются в соответствии с таблицей 1

s1 = r1 ⊕ i1 ⊕ i2 ⊕ i4 = 1 ⊕ 1 ⊕ 0 ⊕ 0 = 0

s2 = r2 ⊕ i1 ⊕ i3 ⊕ i4 = 0 ⊕ 1 ⊕ 0 ⊕ 0 = 1

s3 = r3 ⊕ i2 ⊕ i3 ⊕ i4 = 0 ⊕ 0 ⊕ 0 ⊕ 0 = 0

Синдром S (s1,s2,s3) = 010

Позиция искаженного бита равна десятичному представлению перевернутого числа полученного числа синдрома S:

Искаженный бит: 0102 = 2 позиция

Значит имеется ошибка в r2

Тогда исходное правильное сообщение: 1000

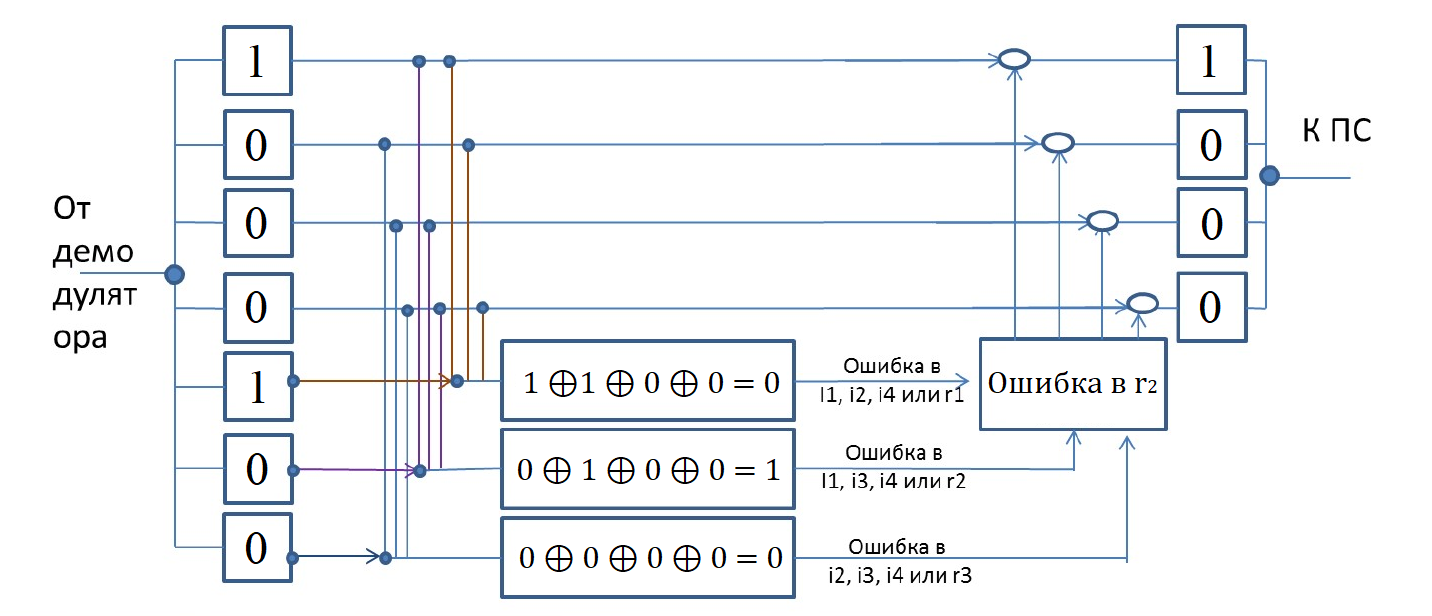
Схема Хемминга для данного случая представлена на рисунке 1

Рисунок 1 – Схема Хемминга для примера 1

Аналогичные расчёты проводятся для остальных последовательностей

**Последовательность № 2**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **r1** | **r2** | **i1** | **r3** | **i2** | **i3** | **i4** |
| 0 | 1 | 0 | 1 | 0 | 0 | 1 |

s1 = r1 ⊕ i1 ⊕ i2 ⊕ i4 = 0 ⊕ 0 ⊕ 0 ⊕ 1 = 1

s2 = r2 ⊕ i1 ⊕ i3 ⊕ i4 = 1 ⊕ 0 ⊕ 0 ⊕ 1 = 0

s3 = r3 ⊕ i2 ⊕ i3 ⊕ i4 = 1 ⊕ 0 ⊕ 0 ⊕ 1 = 0

Синдром S (s1,s2,s3) = 100

Искаженный бит: 0012 = 1 позиция

Ошибка в r1

Тогда исходное правильное сообщение: 0001

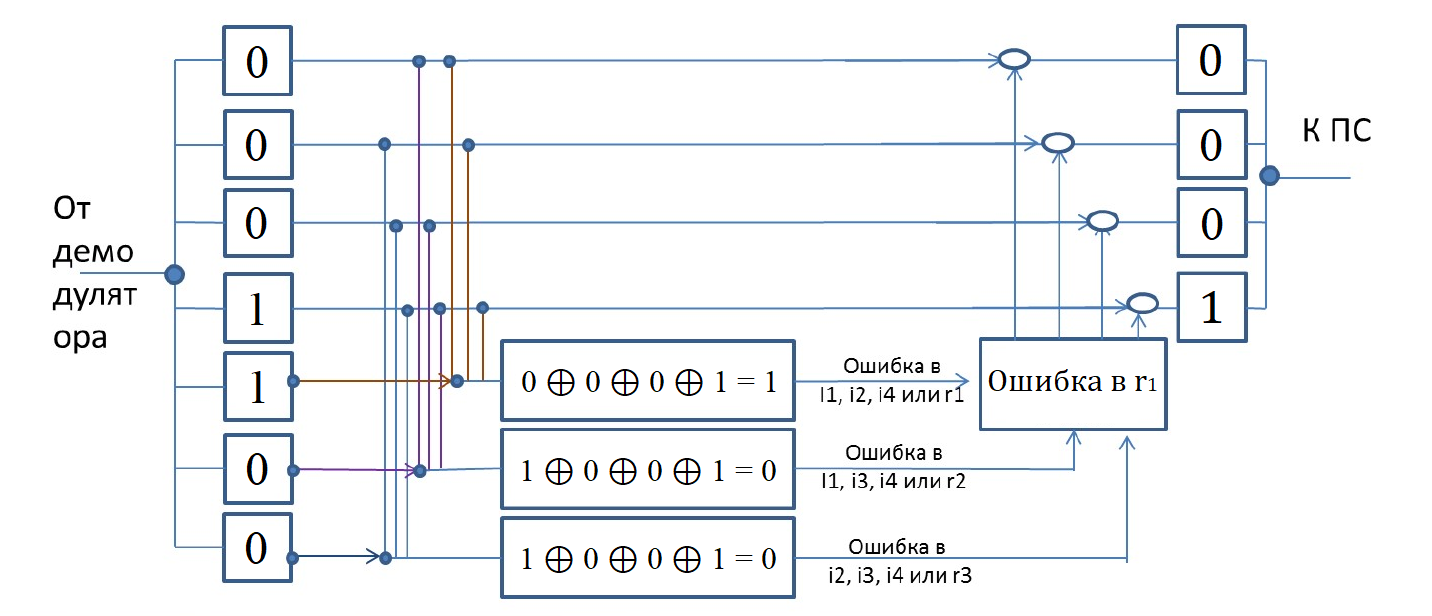
Схема Хемминга для данного случая представлена на рисунке 2

Рисунок 2 - Схема Хемминга для примера 2

**Последовательность № 3**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **r1** | **r2** | **i1** | **r3** | **i2** | **i3** | **i4** |
| 0 | 0 | 0 | 0 | 1 | 0 | 0 |

s1 = r1 ⊕ i1 ⊕ i2 ⊕ i4 = 0 ⊕ 0 ⊕ 1 ⊕ 0 = 1

s2 = r2 ⊕ i1 ⊕ i3 ⊕ i4 = 0 ⊕ 0 ⊕ 0 ⊕ 0 = 0

s3 = r3 ⊕ i2 ⊕ i3 ⊕ i4 = 0 ⊕ 1 ⊕ 0 ⊕ 0 = 1

Синдром S (s1,s2,s3) = 101

Искаженный бит: 1012 = 5 позиция

Ошибка в i2

Тогда исходное правильное сообщение: 0000

Схема Хемминга для данного случая представлена на рисунке 3

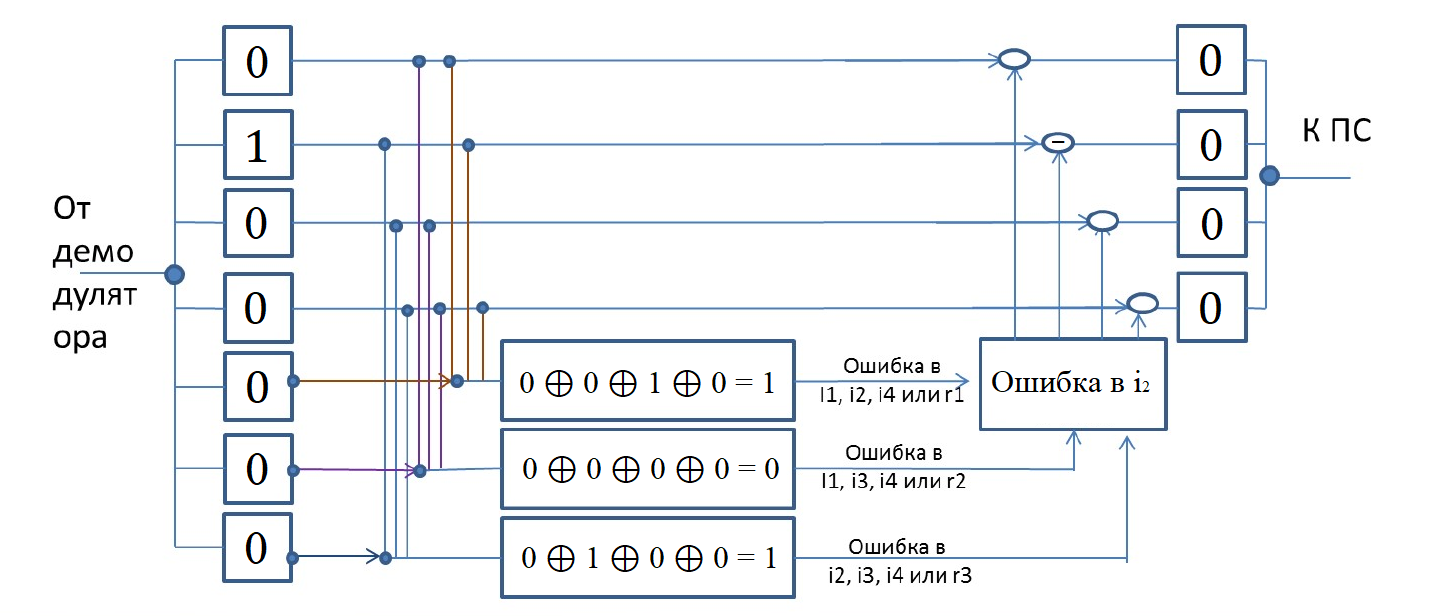


Рисунок 3 – Схема Хемминга для примера 3

**Последовательность № 4**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **r1** | **r2** | **i1** | **r3** | **i2** | **i3** | **i4** |
| 1 | 0 | 1 | 0 | 1 | 1 | 0 |

s1 = r1 ⊕ i1 ⊕ i2 ⊕ i4 = 1 ⊕ 1 ⊕ 1 ⊕ 0 = 1

s2 = r2 ⊕ i1 ⊕ i3 ⊕ i4 = 0 ⊕ 1 ⊕ 1 ⊕ 0 = 0

s3 = r3 ⊕ i2 ⊕ i3 ⊕ i4 = 0 ⊕ 1 ⊕ 1 ⊕ 0 = 0

Синдром S (s1,s2,s3) = 100

Искаженный бит: 0012 = 1 позиция

Ошибка в r1

Тогда исходное правильное сообщение: 1110

Схема Хемминга для данного случая представлена на рисунке 4

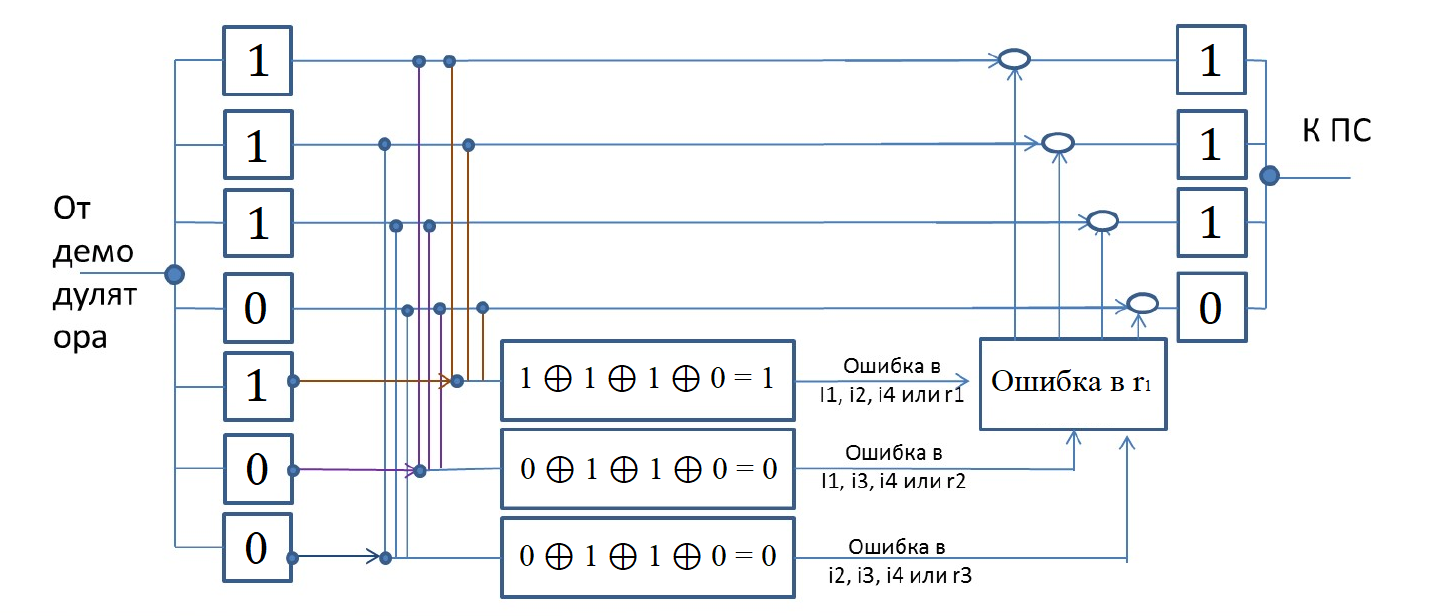


Рисунок 4 – Схема Хемминга для примера 4

## Задание 2

Построить схему декодирования классического кода Хэмминга (15;11), которую представить в отчёте в виде изображения. Показать, исходя из выбранного варианта сообщений, имеются ли в принятом сообщении ошибки, и если имеются, то какие. **Подробно прокомментировать** и записать правильное сообщение.

Входные данные:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **r1** | **r2** | **i1** | **r3** | **i2** | **i3** | **i4** | **r4** | **i5** | **i6** | **i7** | **i8** | **i9** | **i10** | **i11** |
| 0 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 0 |

Таблица 2 – Таблица кодирования информационных битов

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** | **11** | **12** | **13** | **14** | **15** |  |
| **2n** | *r1* | *r2* | *i1* | *r3* | *i2* | *i3* | *i4* | *r4* | *i5* | *i6* | *i7* | *i8* | *i9* | *i10* | *i11* | S |
| **1** | x |  | x |  | x |  | x |  | x |  | x |  | x |  | x | *s1* |
| **2** |  | x | x |  |  | x | x |  |  | x | x |  |  | x | x | *s2* |
| **4** |  |  |  | x | x | x | x |  |  |  |  | x | x | x | x | *s3* |
| **8** |  |  |  |  |  |  |  | x | x | x | x | x | x | x | x | *s4* |

s1 = r1 ⊕ i1 ⊕ i2 ⊕ i4 ⊕ i5 ⊕ i7 ⊕ i9 ⊕ i11 = 0 ⊕ 1 ⊕ 1 ⊕ 0 ⊕ 0 ⊕ 0 ⊕ 1 ⊕ 0 = 1

s2 = r2 ⊕ i1 ⊕ i3 ⊕ i4 ⊕ i6 ⊕ i7 ⊕ i10 ⊕ i11 = 0 ⊕ 1 ⊕ 0 ⊕ 0 ⊕ 1 ⊕ 0 ⊕ 0 ⊕ 0 = 0

s3 = r3 ⊕ i2 ⊕ i3 ⊕ i4 ⊕ i8 ⊕ i9 ⊕ i10 ⊕ i11 = 1 ⊕ 1 ⊕ 0 ⊕ 0 ⊕ 0 ⊕ 1 ⊕ 0 ⊕ 0 = 1

s4 = r4 ⊕ i5 ⊕ i6 ⊕ i7 ⊕ i8 ⊕ i9 ⊕ i10 ⊕ i11 = 1 ⊕ 0 ⊕ 1 ⊕ 0 ⊕ 0 ⊕ 1 ⊕ 0⊕ 0 = 1

Синдром S (s1,s2,s3, s4) = 1011

Искаженный бит: 11012 = 13 позиция

Ошибка в i9

Тогда исходное правильное сообщение: 11000100000

Схема Хемминга для данного случая представлена на рисунке 5

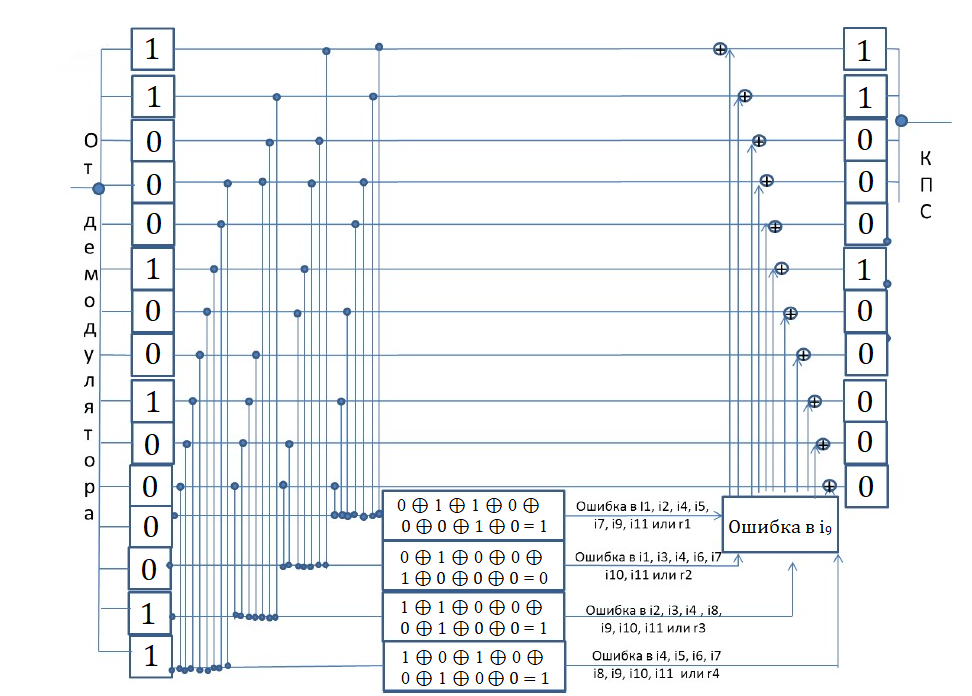


Рисунок 5 – схема Хемминга для набора из 15 бит

## Задание 3

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.

Число информационных разрядов = (57 + 94 + 19 + 10 + 74) \* 4 = 1016

2k = 2048 ≥ 11 + 1016 + 1

В данном случае i = 1016, r = 11, n = 1027

Коэффициент избыточности = r / n = 11 / 1027 ≈ 0,0107

## Задание\*

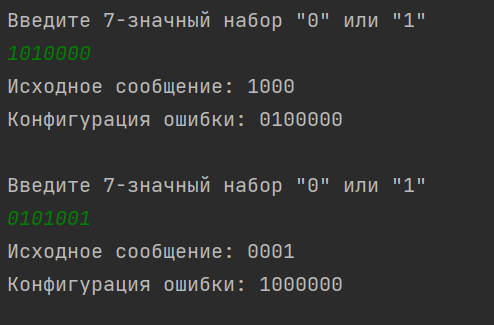
Необязательное задания для получения оценки «5». Написать программу на любом языке программирования, которая на вход из командной строки получает набор из 7 цифр «0» и «1», записанных подряд, анализирует это сообщение на основе классического кода Хэмминга (7,4), а затем выдает правильное сообщение (только информационные биты) и указывает бит с ошибкой при его наличии.

### Исходный код на Java

import java.util.Scanner;  
import java.util.regex.Pattern;  
  
public class HammingCode {  
 public static void main(String[] args) {  
 int[] bits = new int[7];  
 System.*out*.println("Введите 7-значный набор \"0\" или \"1\"");  
 Scanner sin = new Scanner(System.*in*);  
 String inputData = sin.nextLine();  
 while (!Pattern.*matches*("[01]{7}", inputData)) {  
 System.*out*.println("Неверный формат данных. Введите 7-значный набор \"0\" или \"1\"");  
 inputData = sin.nextLine();  
 }  
 for (int i = 0; i < inputData.length(); ++i) {  
 bits[i] = inputData.charAt(i) - '0';  
 }  
 int s1 = *xor*(bits[0], bits[2], bits[4], bits[6]), s2 = *xor*(bits[1], bits[2], bits[5], bits[6]), s3 = *xor*(bits[3], bits[4], bits[5], bits[6]);  
 int position = s1 + 2 \* s2 + 4 \* s3;  
 if (position == 0) {  
 System.*out*.println("Исходное сообщение: " + bits[2] + bits[4] + bits[5] + bits[6] + "\nОшибок нет");  
 } else {  
 StringBuilder errPos = new StringBuilder("0000000");  
 errPos.setCharAt(position - 1, '1');  
 bits[position - 1] = *xor* (bits[position - 1], 1);  
 System.*out*.println("Исходное сообщение: " + bits[2] + bits[4] + bits[5] + bits[6] + "\nКонфигурация ошибки: " + errPos.toString());  
 }  
 }  
  
 public static int xor (int a, int b, int c, int d) {  
 return (a + b + c + d) % 2;  
 }  
 public static int xor (int a, int b) {  
 return (a + b) % 2;  
 }  
}

### Результат работы программы

Примеры взяты из задания №1. Результаты вывода и обработки некорректных входных данных представлены на рисунках 6, 7 и 8.

 Изображение выглядит как текст

Автоматически созданное описание

Рисунок 6 - Примеры 10 и 19

Рисунок 7 - Примеры 57 и 94

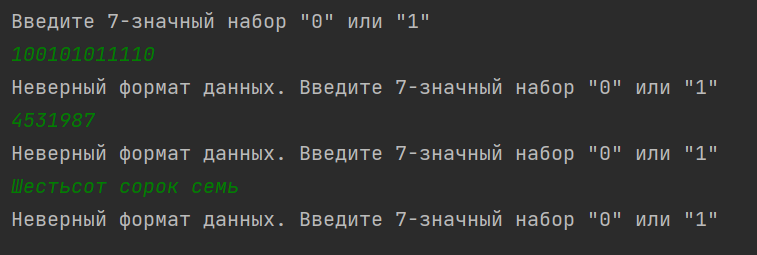


Рисунок 8 - Обработка некорректного ввода

## Вывод

При выполнении данной лабораторной работы я углубился в понятия помехоустойчивого кода, научился анализировать схемы декодирования классического кода Хэмминга (7; 4) и (15;11), сроить таблицы кодирования информационных битов и на основе полученных данных находить ошибки в принятых сообщениях. Отличным упражнением для закрепления материала было написание программы анализатора классического набора Хемминга.

## Список литературы.

1. Черновик методического пособия «Информатика»

<https://vk.com/doc-31201840_566998093>

1. Балакшин П.В., Соснин В.В., Калинин И.В., Малышева Т.А., Раков С.В., Рущенко Н.Г., Дергачев А.М. Информатика: лабораторные работы и тесты: Учебно-методическое пособие

<https://books.ifmo.ru/book/2248/informatika:_laboratornye_raboty_i_testy:_uchebno-metodicheskoe_posobie_/_recenzent:_polyakov_v.i..htm>