

Мегафакультет компьютерных управлений и технологий

Факультет ПИ и КТ

Лабораторная работа №2 по ИКТ

Вариант 29

**Выполнил:**

Болорболд Аригуун

группа P3111

**Преподаватель:**

Малышева Татьяна Алексеевна

г. Санкт-Петербург

2022 год

Оглавление

1. Задание…………………………………………………………………………….…3
2. Решение к заданиям………………………………………………………………....4
3. Вывод………………………………………………………………………………....8
4. Источники………………………………………………………………………….....9

Порядок выполнения работы

1. Определить свой вариант задания с помощью номера в ISU (он же номер студенческого билета). Вариантом является комбинация 3-й и 5-й цифр. Т.е. если номер в ISU = 123456, то вариант = 35.

2. На основании номера варианта задания выбрать набор из 4 полученных сообщений в виде последовательности 7-символьного кода.

3. Построить схему декодирования классического кода Хэмминга (7;4), которую представить в отчёте в виде изображения.

4. Показать, исходя из выбранных вариантов сообщений (по 4 у каждого – часть №1 в варианте), имеются ли в принятом сообщении ошибки, и если имеются, то какие. Подробно прокомментировать и записать правильное сообщение.

5. На основании номера варианта задания выбрать 1 полученное сообщение в виде последовательности 11-символьного кода.

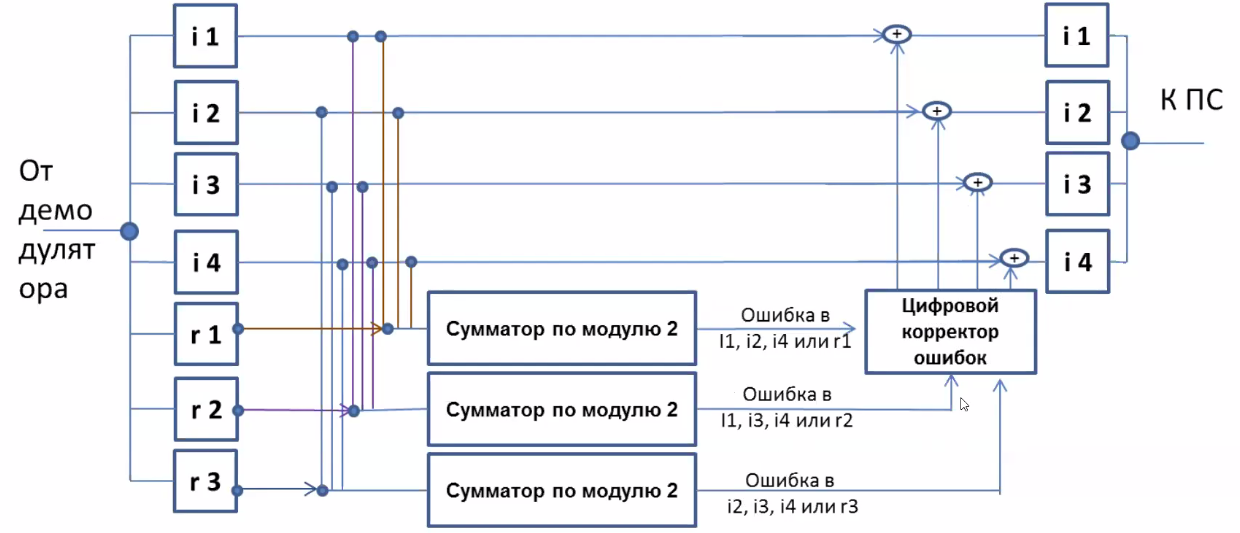
6. Построить схему декодирования классического кода Хэмминга (15;11), которую представить в отчёте в виде изображения.

7. Показать, исходя из выбранного варианта сообщений (по 1 у каждого – часть №2 в варианте), имеются ли в принятом сообщении ошибки, и если имеются, то какие. Подробно прокомментировать и записать правильное сообщение.

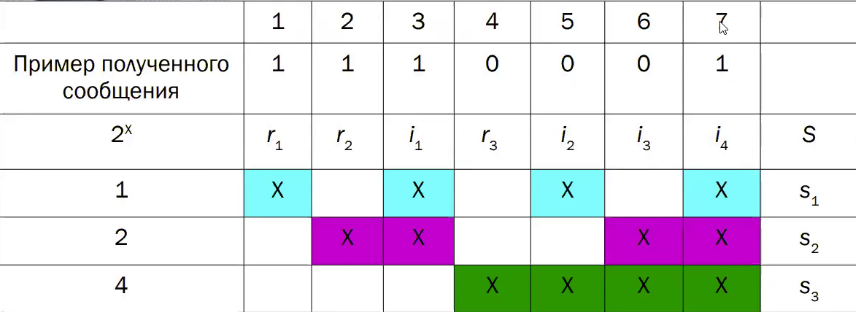
8. Сложить номера всех 5 вариантов заданий. Умножить полученное число на 4. Принять данное число как число информационных разрядов в передаваемом сообщении. Вычислить для данного числа минимальное число проверочных разрядов и коэффициент избыточности.

9. Необязательное задания для получения оценки «5» (позволяет набрать от 86 до 100 процентов от максимального числа баллов БаРС за данную лабораторную). 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.

Схема декодирования классического кода Хэмминга (7, 4):



|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| № | r1 | r2 | i1 | r3 | i2 | ­i3 | i4 |
| 25 | 1 | 0 | 1 | 1 | 0 | 0 | 1 |
| 52 | 1 | 0 | 1 | 1 | 0 | 1 | 1 |
| 79 | 1 | 0 | 0 | 1 | 1 | 0 | 1 |
| 106 | 1 | 0 | 0 | 0 | 1 | 1 | 1 |



25) s1=r1⊕i1⊕i2⊕i4=1+1+0+1=1;

s2=r2⊕i1⊕i3⊕i4=0+1+0+1=0;

s3=r3⊕i2⊕i3⊕i4=1+0+0+1=0;

S=100 => Ошибка в бите r1 => сообщение: 1011011

52) s1=r1⊕i1⊕i2⊕i4=1+1+0+1=1;

s2=r2⊕i1⊕i3⊕i4=0+1+1+1=1;

s3=r3⊕i2⊕i3⊕i4=1+0+1+1=1;

S=111 => Ошибка в бите i4. => сообщение: 1011010

79) s1=r1⊕i1⊕i2⊕i4=1+0+1+1=1;

s2=r2⊕i1⊕i3⊕i4=0+0+0+1=1;

s3=r3⊕i2⊕i3⊕i4=1+1+0+1=1;

S=111 => Ошибка в бите i4 => сообщение: 1011010

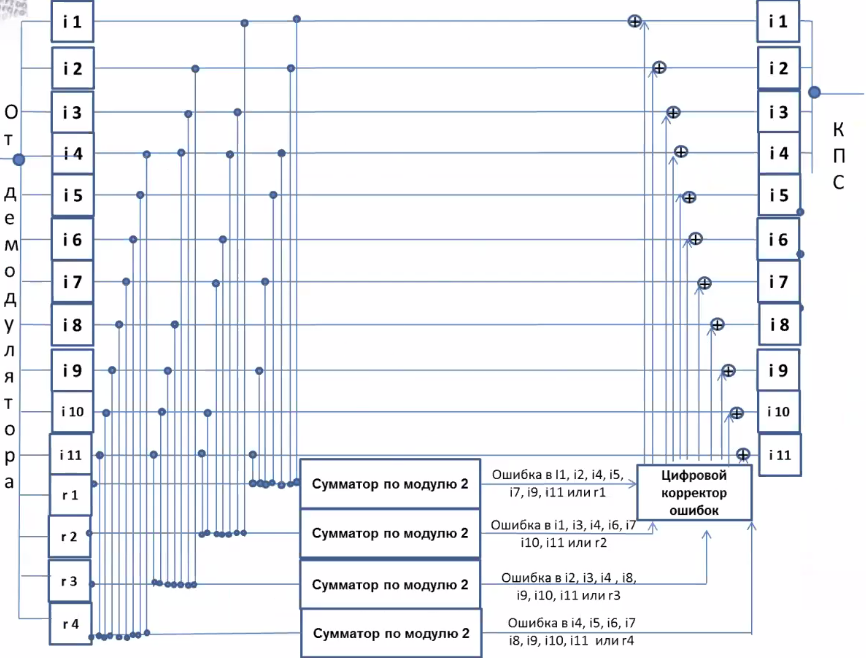
106) s1=r1⊕i1⊕i2⊕i4=1+0+1+1=1;

s2=r2⊕i1⊕i3⊕i4=0+0+1+1=0;

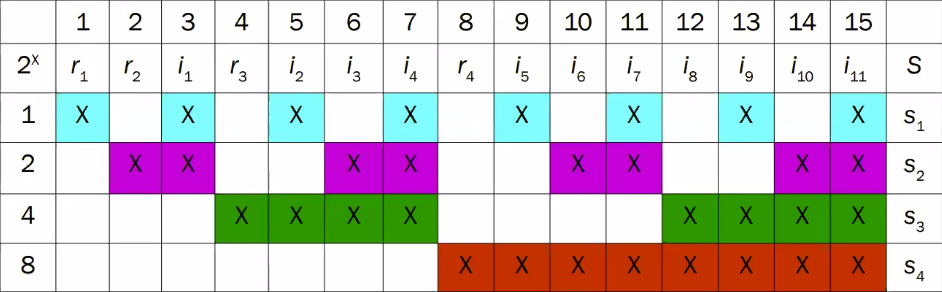
s3=r3⊕i2⊕i3⊕i4=0+1+1+1=1;

S=101 => Ошибка в бите i2 => сообщение: 1011111

Схема декодирования классического кода Хэмминга (15, 11):



|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| № | r1 | r2 | i1 | r3 | i2 | i3 | i4 | r4 | i5 | i6 | i7 | i8 | i9 | i10 | i11 |
| 60 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 |



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

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

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

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

S=1100 => Ошибка в бите i1 => сообщение: 011001111000011

8. i=(25+52+79+106+60)\*4=1288

2r≥r+i+1

2r >=r+12+1

2r >=r+13

32>=18

2r≥r+1288+1=r+1289

Если r=10, то:

210≤1299 == ложь

Если r=11, то:

211≥1300

r=11

Скрипт для кода Хэмминга (на Java, потому что я оборотень):

import java.util.Scanner;  
import java.util.regex.Matcher;  
import java.util.regex.Pattern;  
public class HammingCode {  
 public static void main(String[] args) {  
 String code;  
 do {  
 System.*out*.println("Введите код: ");  
 Scanner sc = new Scanner(System.*in*);  
 code = sc.next();  
 Pattern bin7 = Pattern.*compile*("^[01]+$");  
 Pattern amount = Pattern.*compile*("^.{7}$");  
 Matcher binVer = bin7.matcher(code);  
 Matcher amountVer = amount.matcher(code);  
 boolean binFormat = binVer.find();  
 boolean amountFormat = amountVer.find();  
 if (binFormat) {  
 if (amountFormat) {  
 char[] input = code.toCharArray();  
 byte[] chArr = new byte[input.length];  
 for (int i = 0; i <= input.length - 1; i++) {  
 chArr[i] = (byte) Character.*getNumericValue*(input[i]);  
 }  
 boolean[] b = new boolean[chArr.length];  
 for (int i = 0; i < chArr.length; i++) {  
 b[i] = chArr[i] != 0;  
 }  
 boolean s1 = b[0] ^ b[2] ^ b[4] ^ b[5];  
 boolean s2 = b[1] ^ b[2] ^ b[5] ^ b[6];  
 boolean s3 = b[3] ^ b[4] ^ b[5] ^ b[6];  
 int s1i = 4 \* ((s3) ? 1 : 0) + 2 \* ((s2) ? 1 : 0) + ((s1) ? 1 : 0) - 1;  
 if (s1i == -1) {  
 System.*out*.println("Нет ошибок.");  
 } else {  
 b[s1i] = !b[s1i];  
 int i1f = b[2] ? 1 : 0;  
 int i2f = b[4] ? 1 : 0;  
 int i3f = b[5] ? 1 : 0;  
 int i4f = b[6] ? 1 : 0;  
 System.*out*.println("Исправленное сообщение: " + " " + i1f + " " + i2f + " " + i3f + " " + i4f);  
 }  
 } else {  
 System.*out*.println("Не забыли ли вы, что данная программа принимает точно и только 7 цифр?");  
 }  
 } else {  
 System.*out*.println("Зачем творить дичь? Формат ввода - двоичная.");  
 }  
 } while (!code.equals("stop"));  
 }  
}

Скрипт для кода Хэмминга (на Python, потому что я ущемился и переделал всё):

import re  
while True:  
 codeInput = input("Введите код: ")  
 if codeInput == "stop":  
 break  
 else:  
 bin7 = re.compile("^[01]+$")  
 if re.findall(bin7, codeInput):  
 binPlace = re.compile("^[01]{7}$")  
 if re.findall(binPlace, codeInput):  
 codeList = [int(b) for b in str(codeInput)]  
 r1 = bool(codeList[0])  
 r2 = bool(codeList[1])  
 i1 = bool(codeList[2])  
 r3 = bool(codeList[3])  
 i2 = bool(codeList[4])  
 i3 = bool(codeList[5])  
 i4 = bool(codeList[6])  
 boolList = [r1, r2, i1, r3, i2, i3, i4]  
 s1 = r1 ^ i1 ^ i2 ^ i4  
 s2 = r2 ^ i1 ^ i3 ^ i4  
 s3 = r3 ^ i2 ^ i3 ^ i4  
 s1i = int(s1)  
 s2i = int(s2)  
 s3i = int(s3)  
 notIndex = (s1i \* 4 + s2i \* 2 + s3i)-1  
 if notIndex == -1:  
 print("Нет ошибок. Хорошего дня!")  
 else:  
 boolList[notIndex] = not boolList[notIndex]  
 i1f = int(boolList[2])  
 i2f = int(boolList[4])  
 i3f = int(boolList[5])  
 i4f = int(boolList[6])  
 print("Исправленное сообщение: ", i1f, i2f, i3f, i4f)  
 else:  
 print("Не забыли ли вы, что данная программа принимает точно и только 7 цифр?")  
 else:  
 print("Зачем творить дичь? Формат ввода - двоичная.")

Вывод: на второй лабораторной работе я окончательно овладел методом кодирования Хэмминга, в основном классическим кодом. Если у студента нет примерного понятия фундаментальных определений (то есть я), то данная тема даётся нелегко, но можно разбираться. Заслуга Хэмминга имеет огромное значение, так как этот код и является фундаментом для исправления ошибок в бите.

