ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ

ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«Национальный исследовательский университет ИТМО»

ФАКУЛЬТЕТ ПРОГРАММНОЙ ИНЖЕНЕРИИ И КОМПЬЮТЕРНОЙ ТЕХНИКИ

**ЛАБОРАТОРНАЯ РАБОТА №2**

**«Перевод чисел между различными системами счисления»**

по дисциплине «Информатика»

Вариант № 76

**Группа:** P3112

**Выполнил**: Пархоменко К. А.

**Проверил:** Белозубов А.В.

г. Санкт-Петербург

2022 г.

Оглавление

[Задания 3](#_Toc118722954)

[Задание 1 3](#_Toc118722955)

[Задание 2 3](#_Toc118722956)

[Задание 3 3](#_Toc118722957)

[Задание 4 3](#_Toc118722958)

[Задание 5 5](#_Toc118722959)

[Задание 6 6](#_Toc118722960)

[Задание 7 6](#_Toc118722961)

[Задание 8 7](#_Toc118722962)

[Дополнительное задание 7](#_Toc118722963)

[Заключение 8](#_Toc118722964)

[Список литературы 9](#_Toc118722965)

# Задания

# Задание 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 = 12**3**4**5**6, то вариант = 35.

Решение:

Табельный номер – 36**7**4**6**8

Вариант – 76

# Задание 2

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

Решение:

|  |  |  |  |
| --- | --- | --- | --- |
| 58 | 95 | 20 | 10 |

# 

# Задание 3

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

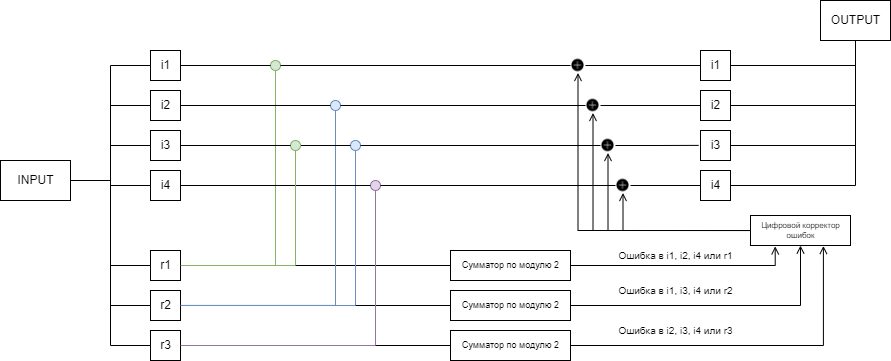


Рисунок 1 - схема Хэмминга (7;4)

# Задание 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.

Вариант 58:

Полученное сообщение: **0001100**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 1 | 2 | 3 | 4 | 5 | 6 | 7 |  |
| 2x | r1 | r2 | i1 | r3 | i2 | i3 | i4 | s |
| 1 | **0** | 0 | **0** | 1 | **1** | 0 | **0** | 1 |
| 2 | 0 | **0** | **0** | 1 | 1 | **0** | **0** | 0 |
| 4 | 0 | 0 | 0 | **1** | **1** | **0** | **0** | 0 |

**Этапы вычисления:**

1. Определяем синдром последовательности по следующим формулам:

s1 = r1 ⊕ i1 ⊕ i2 ⊕ i4

s2 = r2 ⊕ i1 ⊕ i3 ⊕ i4

s3 = r3 ⊕ i2 ⊕ i3 ⊕ i4

s1 = 0 ⊕ 0 ⊕ 1 ⊕ 0 = 1

s2 = 0 ⊕ 0 ⊕ 0 ⊕ 0 = 0

s3 = 1 ⊕ 1 ⊕ 0 ⊕ 0 = 0

1. Получаем 100
2. Переворачиваем 100 и получаем 001
3. Переводим 100 из двоичной системы счисления в десятичную
4. Получаем 1 — номер бита с ошибкой
5. Меняем 1–ый бит на обратный ему (0 на 1).
6. Получаем правильное сообщение

**Правильное сообщение: 1001100**

Вариант 95:

Полученное сообщение: **1011110**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 1 | 2 | 3 | 4 | 5 | 6 | 7 |  |
| 2x | r1 | r2 | i1 | r3 | i2 | i3 | i4 | s |
| 1 | **1** | 0 | **1** | 1 | **1** | 1 | **0** | 1 |
| 2 | 1 | **0** | **1** | 1 | 1 | **1** | **0** | 0 |
| 4 | 1 | 0 | 1 | **1** | **1** | **1** | **0** | 1 |

**Этапы вычисления:**

1. Определяем синдром последовательности по следующим формулам:

s1 = r1 ⊕ i1 ⊕ i2 ⊕ i4

s2 = r2 ⊕ i1 ⊕ i3 ⊕ i4

s3 = r3 ⊕ i2 ⊕ i3 ⊕ i4

s1 = 1 ⊕ 1 ⊕ 1 ⊕ 0 = 1

s2 = 0 ⊕ 1 ⊕ 1 ⊕ 0 = 0

s3 = 1 ⊕ 1 ⊕ 1 ⊕ 0 = 1

1. Получаем 101
2. Переворачиваем 101
3. Получаем 101
4. Переводим 101 из двоичной системы счисления в десятичную
5. Получаем 2 — номер бита с ошибкой
6. Меняем 5–ый бит на обратный ему (1 на 0)
7. Получаем правильное сообщение

**Правильное сообщение: 1011010**

Вариант 20:

Полученное сообщение: **0110001**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 1 | 2 | 3 | 4 | 5 | 6 | 7 |  |
| 2x | r1 | r2 | i1 | r3 | i2 | i3 | i4 | s |
| 1 | **0** | 1 | **1** | 0 | **0** | 0 | **1** | 0 |
| 2 | 0 | **1** | **1** | 0 | 0 | **0** | **1** | 1 |
| 4 | 0 | 1 | 1 | **0** | **0** | **0** | **1** | 1 |

**Этапы вычисления:**

1. Определяем синдром последовательности по следующим формулам:

s1 = r1 ⊕ i1 ⊕ i2 ⊕ i4

s2 = r2 ⊕ i1 ⊕ i3 ⊕ i4

s3 = r3 ⊕ i2 ⊕ i3 ⊕ i4

s1 = 0 ⊕ 1 ⊕ 0 ⊕ 1 = 0

s2 = 1 ⊕ 1 ⊕ 0 ⊕ 1 = 1

s3 = 0 ⊕ 0 ⊕ 0 ⊕ 1 = 1

1. Получаем 011
2. Переворачиваем 011
3. Получаем 110
4. Переводим 110 из двоичной системы счисления в десятичную
5. Получаем 6 — номер бита с ошибкой
6. Меняем 6–ой бит на обратный ему (0 на 1)
7. Получаем правильное сообщение

**Правильное сообщение: 0110011**

Вариант 10:

Полученное сообщение: **1010000**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 1 | 2 | 3 | 4 | 5 | 6 | 7 |  |
| 2x | r1 | r2 | i1 | r3 | i2 | i3 | i4 | s |
| 1 | **1** | 0 | **1** | 0 | **0** | 0 | **0** | 0 |
| 2 | 1 | **0** | **1** | 0 | 0 | **0** | **0** | 1 |
| 4 | 1 | 0 | 1 | **0** | **0** | **0** | **0** | 0 |

**Этапы вычисления:**

1. Определяем синдром последовательности по следующим формулам:

s1 = r1 ⊕ i1 ⊕ i2 ⊕ i4

s2 = r2 ⊕ i1 ⊕ i3 ⊕ i4

s3 = r3 ⊕ i2 ⊕ i3 ⊕ i4

s1 = 1 ⊕ 1 ⊕ 0 ⊕ 0 = 0

s2 = 0 ⊕ 1 ⊕ 0 ⊕ 0 = 1

s3 = 0 ⊕ 0 ⊕ 0 ⊕ 0 = 0

1. Получаем 010
2. Переворачиваем 010
3. Получаем 010
4. Переводим 010 из двоичной системы счисления в десятичную
5. Получаем 2 — номер бита с ошибкой
6. Меняем 2–ой бит на обратный ему (0 на 1)
7. Получаем правильное сообщение

**Правильное сообщение: 1110000**

# Задание 5

На основании номера варианта задания выбрать 1 полученное сообщение в

виде последовательности 11-символьного кода.

Полученное сообщение - **001110010110100**

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

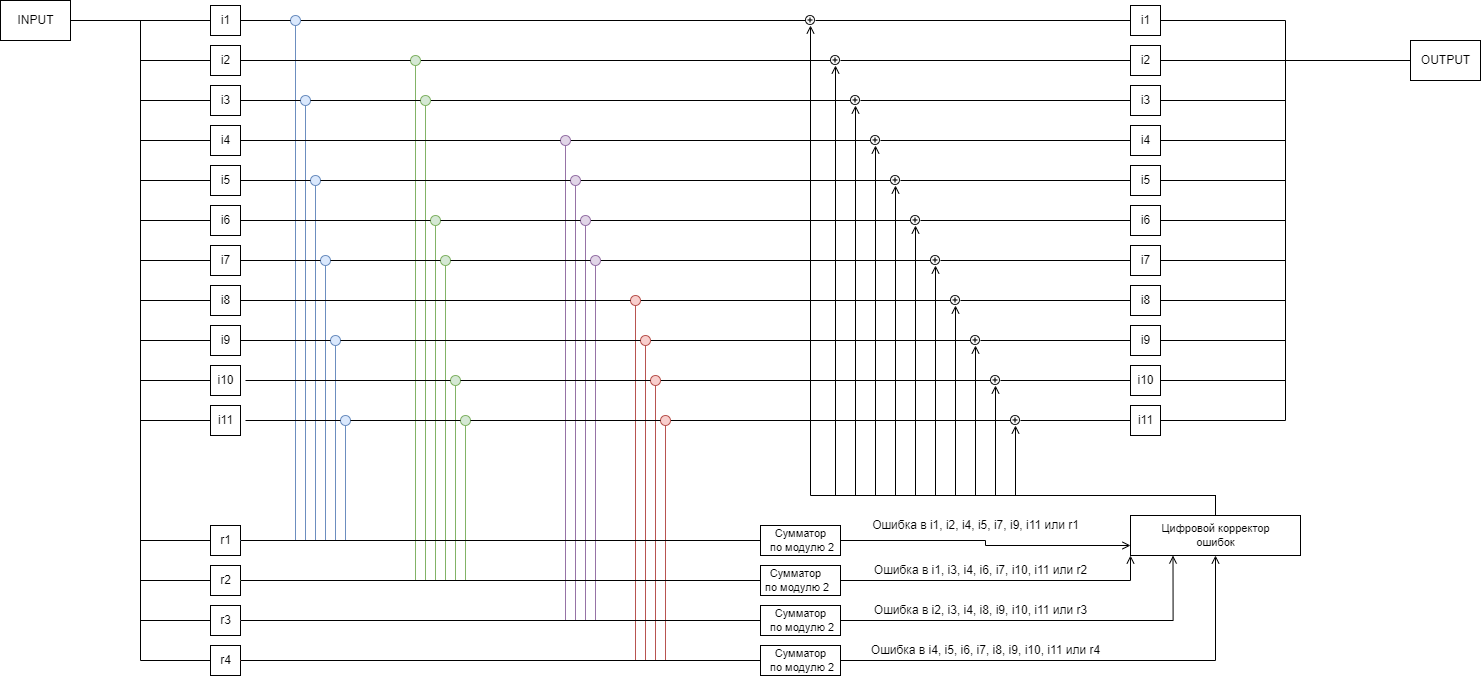


Рисунок 2 - схема Хэмминга (15;11)

# Задание 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.

Вариант 75

Полученное сообщение - **001110010110100**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |  |
| 2x | r1 | r2 | i1 | r3 | i2 | i3 | i4 | r4 | i5 | i6 | i7 | i8 | i9 | i10 | i11 | s |
| 1 | **0** | 0 | **1** | 1 | **1** | 0 | **0** | 1 | **0** | 1 | **1** | 0 | **1** | 0 | **0** | 0 |
| 2 | 0 | **0** | **1** | 1 | 1 | **0** | **0** | 1 | 0 | **1** | **1** | 0 | 1 | **0** | **0** | 1 |
| 4 | 0 | 0 | 1 | **1** | **1** | **0** | **0** | 1 | 0 | 1 | 0 | **0** | **1** | **0** | **0** | 1 |
| 8 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | **1** | **1** | **1** | **0** | **0** | **1** | **0** | **0** | 0 |

**Основные этапы вычисления:**

1. Определяем синдром последовательности:

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

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

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

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

s1 = 0 ⊕ 1 ⊕ 1 ⊕ 0 ⊕ 0 ⊕ 1 ⊕ 1 ⊕ 0 = 0

s2 = 0 ⊕ 1 ⊕ 0 ⊕ 0 ⊕ 1 ⊕ 1 ⊕ 0 ⊕ 0 = 1

s3 = 1 ⊕ 1 ⊕ 0 ⊕ 0 ⊕ 0 ⊕ 1 ⊕ 0 ⊕ 0 = 1

s4 = 1 ⊕ 1 ⊕ 1 ⊕ 0 ⊕ 0 ⊕ 1 ⊕ 0 ⊕ 0 = 0

1. Получаем 0110
2. Переворачиваем 0110
3. Получаем 0110
4. Переводим 0110 из двоичной системы счисления в десятичную
5. Получаем 6 — бит в котором ошибка
6. Меняем 6-ой бит на обратный ему (0 на 1)
7. Получаем правильное сообщение

**Правильное сообщение:** **001111010110100**

# Задание 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.

**Основные этапы вычисления:**

1. Вычисляем число информационных разрядов

(58 + 95 + 20 + 10 + 75) \* 4 = 1032

1. Вычисляем минимальное число проверочных разрядов по формуле

2r ≥ r + i + 1

1. Получаем r = 11
2. Вычисляем коэффициент избыточности по формуле

r / n = r / (r + i)

1. Получаем r/n ≈ 0,0106692

**Ответ:** r = 11, r/n ≈ 0,0106692.

# Дополнительное задание

package main

import (

"fmt"

"log"

"regexp"

"strconv"

)

func main() {

symbolCode := make([]int, 7)

{

var buffString string

fmt.Print("Введите 7-ми символьный код: ")

\_, err := fmt.Scanf("%s", &buffString)

match, \_ := regexp.Match("^[0-1]{7}$", []byte(buffString))

if err != nil || !match {

log.Fatalln("неправильный формат ввода -> " + buffString)

}

for i, v := range buffString {

symbolCode[i], \_ = strconv.Atoi(string(v))

}

}

bitmap := map[int]string{

0: "r1",

1: "r2",

2: "i1",

3: "r3",

4: "i2",

5: "i3",

6: "i4"}

var s1, s2, s3 int

s1 = symbolCode[0] ^ symbolCode[2] ^ symbolCode[4] ^ symbolCode[6]

s2 = symbolCode[1] ^ symbolCode[2] ^ symbolCode[5] ^ symbolCode[6]

s3 = symbolCode[3] ^ symbolCode[4] ^ symbolCode[5] ^ symbolCode[6]

binConvertedToInt, err := strconv.ParseInt(fmt.Sprintf("%d%d%d", s3, s2, s1), 2, 8)

if err != nil {

log.Fatalln("unable to convert bin to int")

}

if binConvertedToInt == 0 {

fmt.Println("Ошибок не обнаружено")

return

}

fmt.Printf("Ошибка в бите с номером %d (%s)\n", binConvertedToInt, bitmap[int(binConvertedToInt-1)])

if symbolCode[int(binConvertedToInt-1)] == 0 {

symbolCode[int(binConvertedToInt-1)] = 1

} else {

symbolCode[int(binConvertedToInt-1)] = 0

}

fmt.Print("Правильное сообщение: ")

for \_, v := range symbolCode {

fmt.Printf("%d", v)

}

}

Программа на языке Go для определения и исправления ошибочных бит

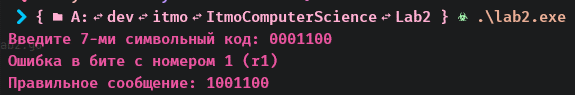


Рисунок 3 – Вывод в терминал

# Заключение

В ходе выполнения лабораторной работы были построены схемы Хэмминга (7;4) и (15;11). Определены и исправлены биты информации переданные ошибочно. Кроме этого, была написана программа на языке **Go** для поиска и исправления бита с ошибкой.

# Список литературы

[В Интернете].

**Код Хэмминга. Пример работы алгоритма** [В Интернете] / авт. tltshnik // habr. - 12 10 2012 г.. - 23 10 2022 г.. - https://habr.com/ru/post/140611/.

**Кодирование с исправлением ошибок** [В Интернете] / авт. Мун Тодд К. // engineering. - 15 4 2005 г.. - 23 10 2022 г.. - https://engineering.usu.edu/.