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

Факультет программной инженерии и компьютерной техники

Лабораторная работа №2

**Синтез помехоустойчивого кода**

Вариант №69

Выполнил

Макогон Ярослав Вадимович

Номер группы: P3118

Проверилa

[Малышева Т. А.](https://my.itmo.ru/persons/165275)

**Содержание**

[Задание 3](#_Toc179460801)

[Основные этапы вычислений 4](#_Toc179460802)

[Заключение 13](#_Toc179460805)

[Список использованных источников 14](#_Toc179460809)

### Задание

Обязательное задание:

1. Построить схему декодирования классического кода Хэмминга (7;4). Показать имеются ли в принятом сообщении ошибки, и если имеются, то какие. Подробно прокомментировать и записать правильное сообщение:
   1. 51 - 1010011
   2. 88 - 0100110
   3. 13 - 1101000
   4. 10 - 1010000
2. Построить схему декодирования классического кода Хэмминга (15;11), которую представить в отчёте в виде изображения. Показать имеются ли в принятом сообщении ошибки, и если имеются, то какие. Подробно прокомментировать и записать правильное сообщение.
   1. 69 - 001110001010100
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.

Дополнительное задание:

Написать программу на любом языке программирования, которая на вход 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 вычислений

Обязательное задание:

### Схема декодирования для классического кода Хэмминга (7;4):

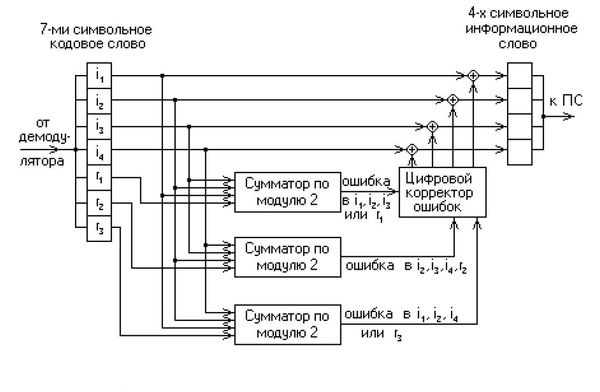


Рисунок 1.Схема декодирования для классического кода Хэмминга (7;4)

### Схема декодирования для классического кода Хэмминга (15;11):

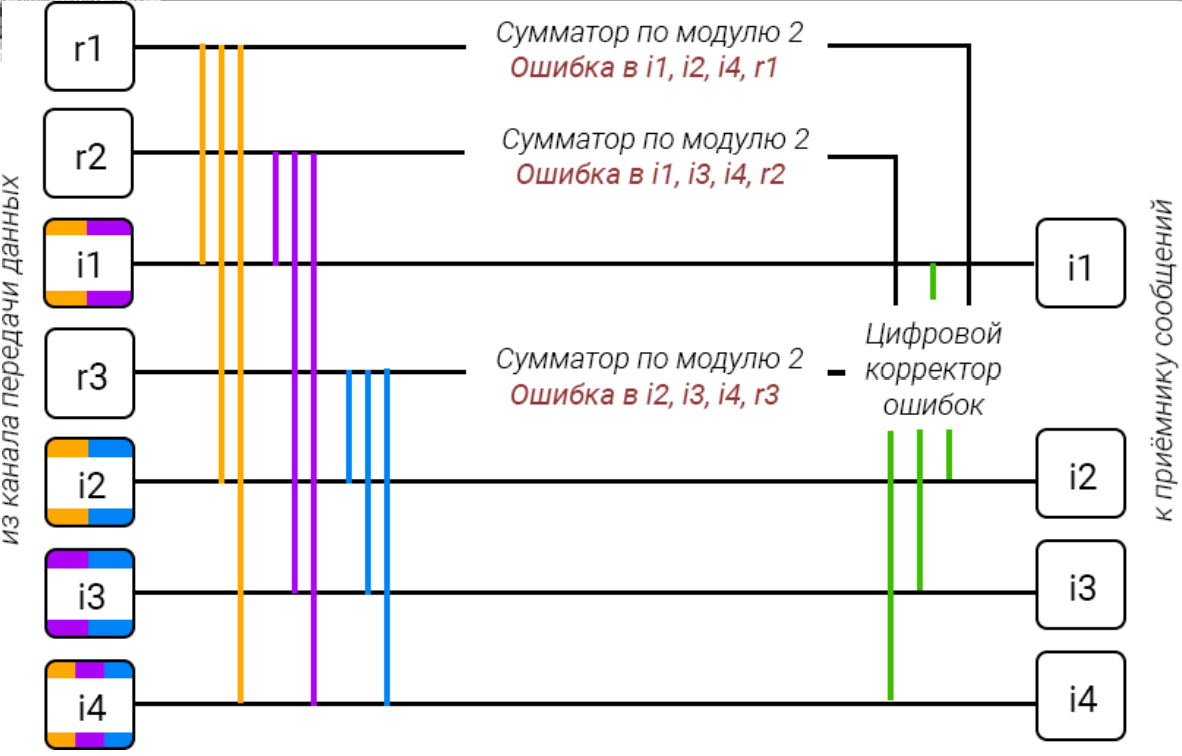


Рисунок 2. Схема декодирования для классического кода Хэмминга (15;11) - Часть 1

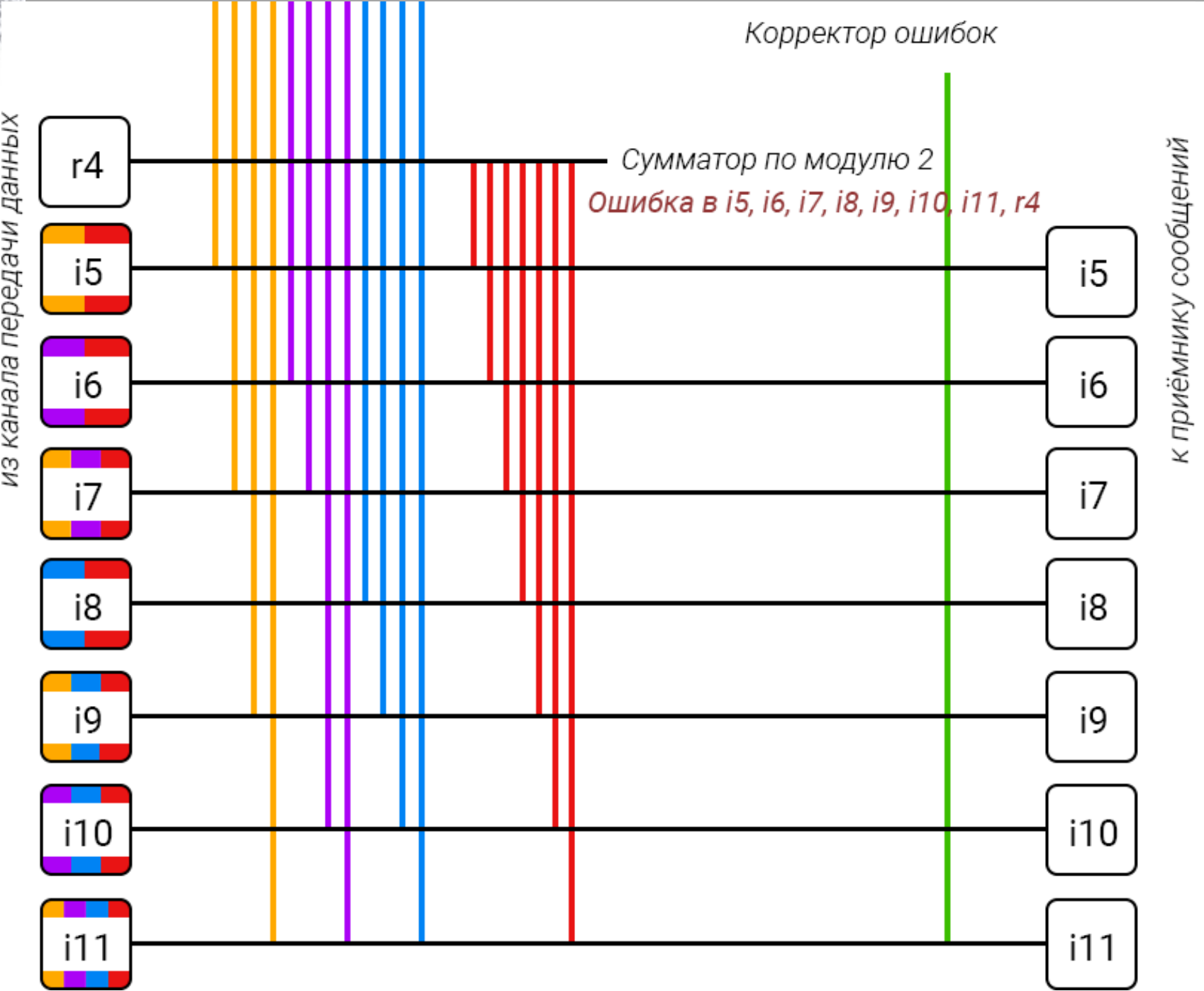


Рисунок 3. Схема декодирования для классического кода Хэмминга (15;11) - Часть 2

**Задание №1**

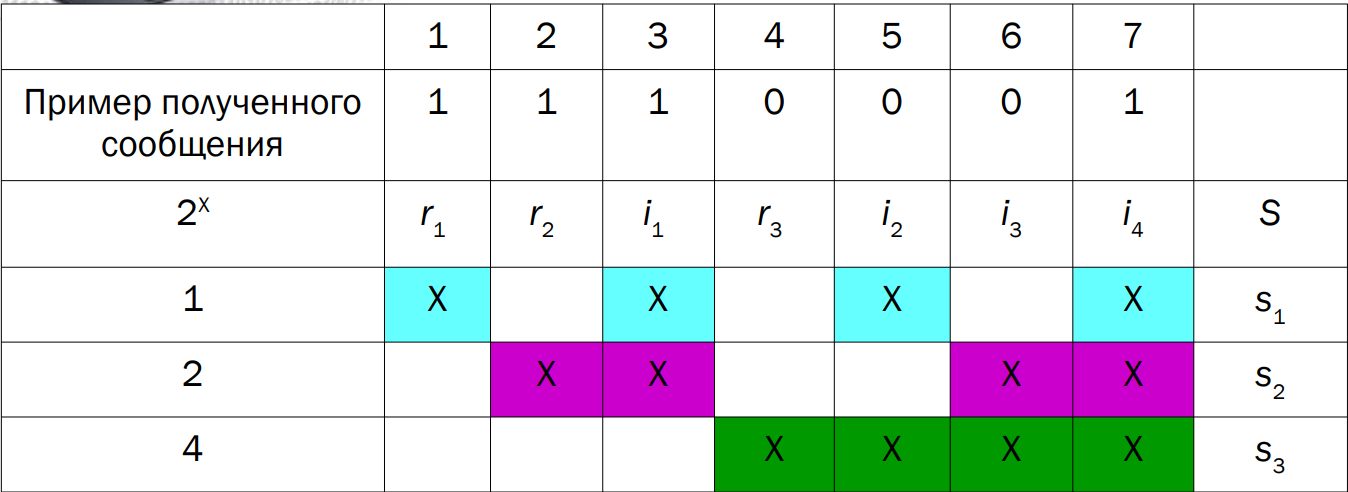
* 1. 51 – 1010011

Рисунок 4. Таблица кода Хэмминга (7;4)

, значит ошибка в .

Исходное сообщение: 1000011

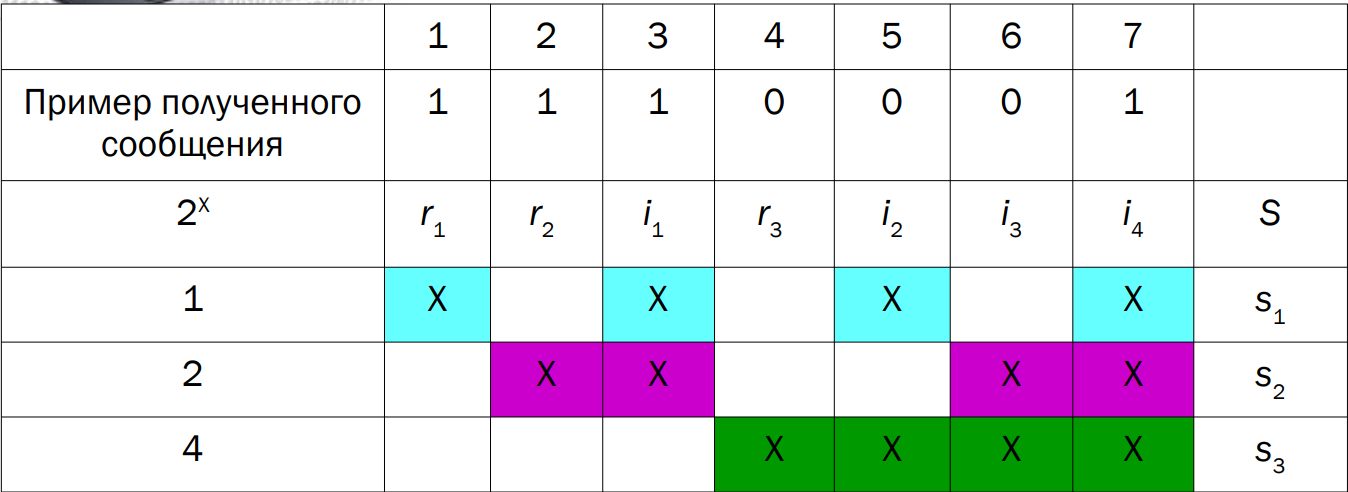
* 1. 88 – 0100110

Рисунок 5. Таблица кода Хэмминга (7;4)

, значит ошибка в .

Исходное сообщение: 1100110

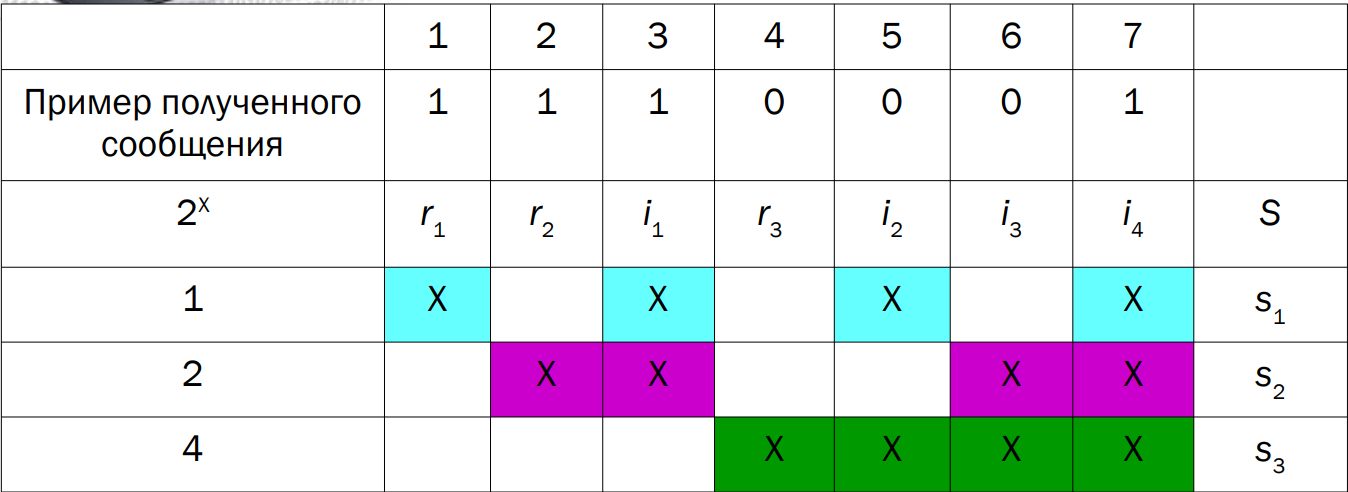
* 1. 13 - 1101000

Рисунок 6. Таблица кода Хэмминга (7;4)

, значит ошибка в .

Исходное сообщение: 1101001

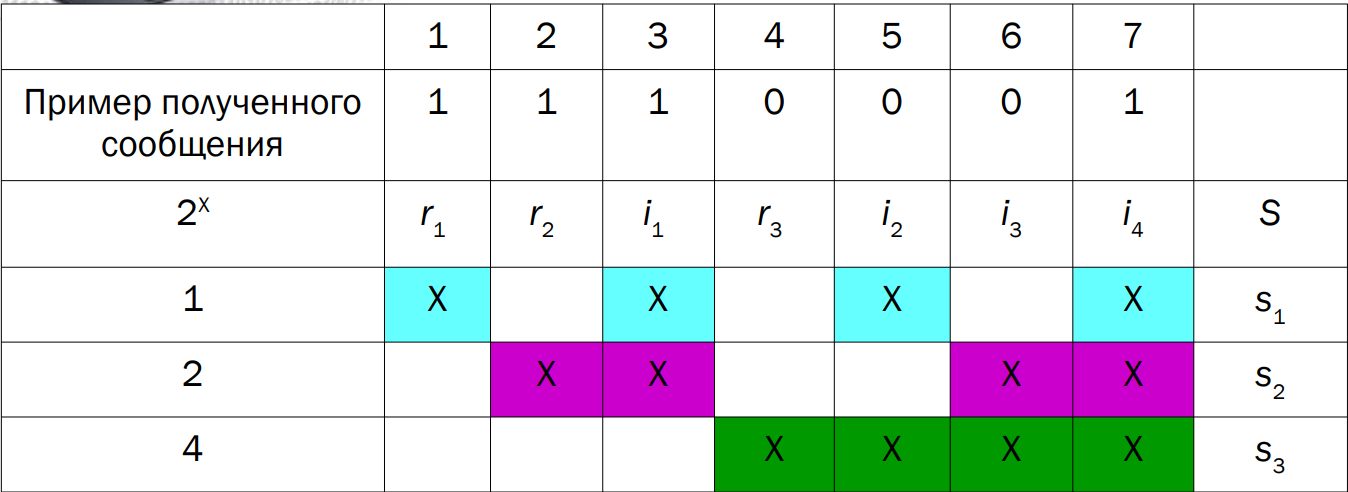
* 1. 10 - 1010000

Рисунок 7. Таблица кода Хэмминга (7;4)

, значит ошибка в .

Исходное сообщение: 1110000

**Задание №2**

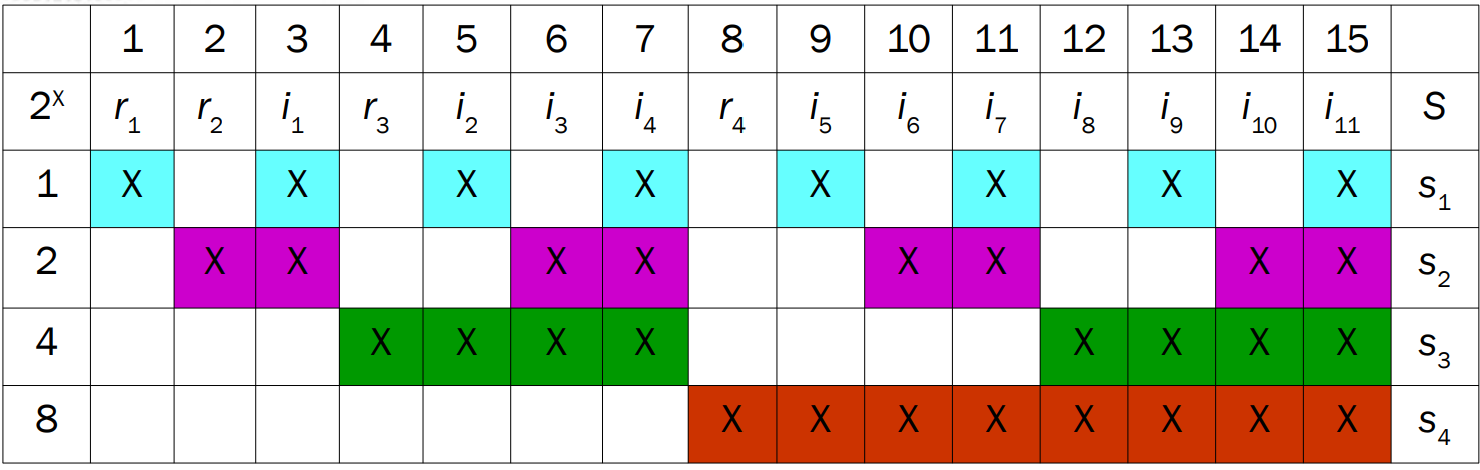
1. 69 - 001110001010100

Рисунок 8. Таблица кода Хэмминга (15;11)

, значит ошибка в .

Исходное сообщение: 001110001010000

**Задание №3**

*(51 + 88 + 13 + 10 + 69) \* 4 = 924 => i = 924 -* число информационных разрядов

*=> -* минимальное число проверочных разрядов

k = - коэффициент избыточности

Дополнительное задание:

**C++17**

#include <iostream>

#include <string>

**void** **drop\_input\_error**() {

std::cout << "Incorrect input format.";

exit(**0**);

}

**signed** **main**() {

std::cout << "Enter a message: ";

std::string input;

std::cin >> input;

**if** (input.size() != **7**) {

drop\_input\_error();

}

**bool** message[**7**];

**for** (**int** i = **0**; i < **7**; ++i) {

**if** (input[i] != '0' && input[i] != '1') {

drop\_input\_error();

}

message[i] = (input[i] == '0' ? false : true);

}

**bool** s1 = message[**0**] xor message[**2**] xor message[**4**] xor message[**6**],

s2 = message[**1**] xor message[**2**] xor message[**5**] xor message[**6**],

s3 = message[**3**] xor message[**4**] xor message[**5**] xor message[**6**];

**int** mistake\_index = s1 + s2 \* **2** + s3 \* **4** - **1**;

std::string correct\_message = input;

**if** (mistake\_index == -**1**) {

std::cout << "Message is correct.**\n**";

}

**else** {

std::string output\_list[**7**] = { "r\_1","r\_2","i\_1","r\_3","i\_2","i\_3","i\_4" };

correct\_message[mistake\_index] = (correct\_message[mistake\_index] == '1' ? '0' : '1');

std::cout << "Error in " << output\_list[mistake\_index] << ".**\n**Index of incorrect bit in message: " << mistake\_index + **1** << '\n';

}

std::string output;

output.push\_back(correct\_message[**2**]);

output.push\_back(correct\_message[**4**]);

output.push\_back(correct\_message[**5**]);

output.push\_back(correct\_message[**6**]);

std::cout << "Information bits in correct message: " << output;

}

**Пример вывода программы:**

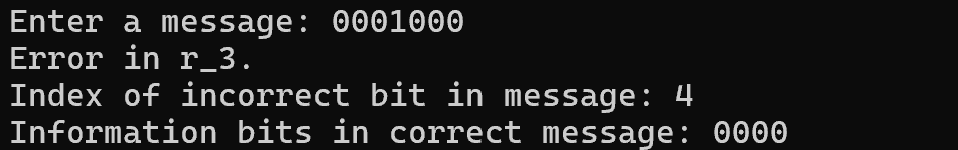


Рисунок 9. Пример работы программы

### Заключение

### Был изучен принцип работы кода Хэмминга.

### Проведены вычисления для выявления ошибок в сообщениях разной длинны.

### Написана программа на языке программирования C++, которая выявляет ошибку в сообщении, анализируя его по на основе классического кода Хэмминга (7,4), а затем выдает правильное сообщение (только информационные биты) и указывает бит с ошибкой при его наличии.

### Список использованных источников

1. Статья “Код Хэмминга. Пример работы алгоритма” – URL: <https://habr.com/ru/articles/140611/>
2. Статья “Избыточное кодирование, код Хэмминга” – URL: <https://clck.ru/3DozQW>