МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «Санкт-Петербургский национальный исследовательский университет ИТМО»

ФАКУЛЬТЕТ ПРОГРАММНОЙ ИНЖЕНЕРИИ И КОМПЬЮТЕРНОЙ ТЕХНИКИ

**ЛАБОРАТОРНАЯ РАБОТА No2**

по дисциплине

«ИНФОРМАТИКА»

Вариант № 70

***Выполнил:***

Студент группы P3116

Билошицкий Михаил Владимирович

***Преподаватель:***

Машина Екатерина Алексеевна

Санкт-Петербург, 2022

# Содержание

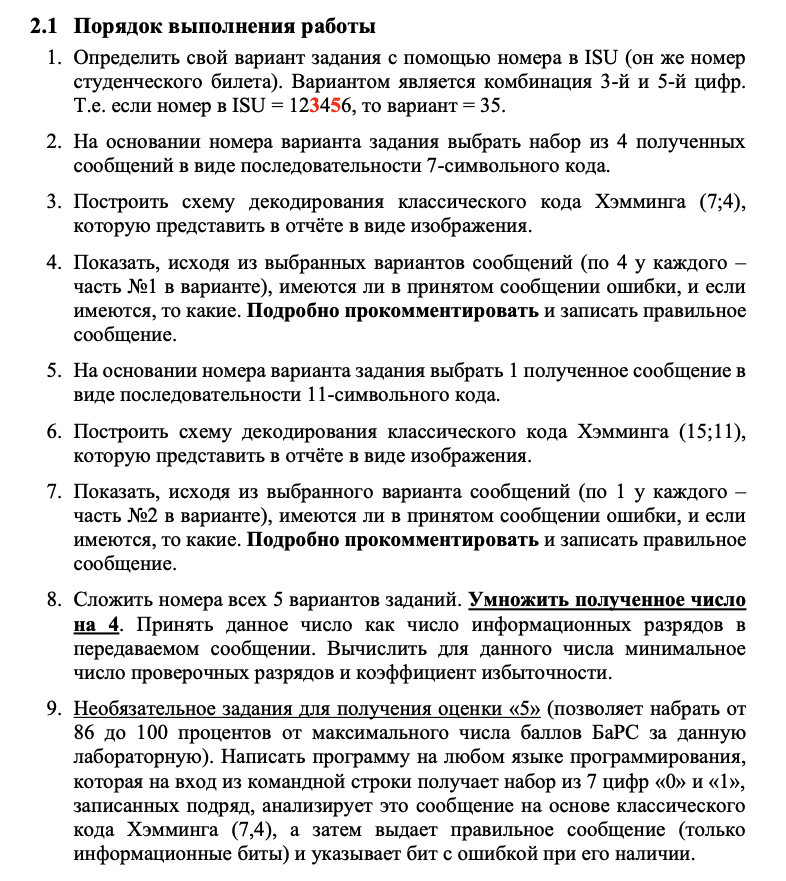
[Задания 3](#_Toc115089178)

[Основные этапы вычисления 5](#_Toc115089179)

[Вывод 9](#_Toc115089180)

# Задания

Основное условие



Задание 1-2



Задание 1 (коды)









Задание 2 (код)



# Основные этапы вычисления

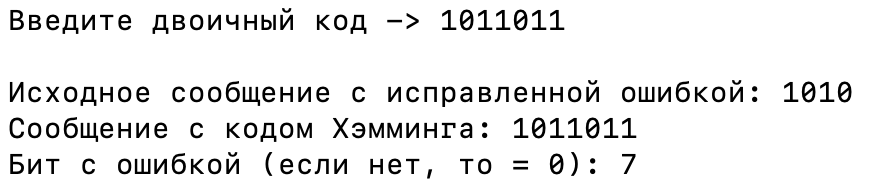
Для вычислений использовалась программа, написанная на языке Python, в которой был реализован алгоритм кодирования в код Хэмминга и Декодирования из него. Программа автоматически выводит информацию с исходным сообщением с исправленной ошибкой и номер ошибочного бита.

Код программы:

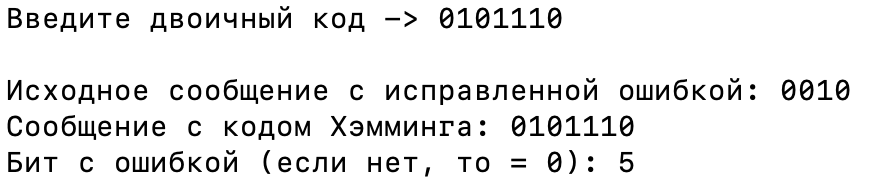
**def** insert\_symbol(my\_str, sybmol, index):  
    **return** my\_str[:index] **+** sybmol **+** my\_str[index:]  
    
**def** replace\_by\_index(my\_str, symbol, index):  
    **return** my\_str[:index] **+** symbol **+** my\_str[index **+** 1:]  
    
# encode\_shred Закодировать сообщение в код Хэмминга  
# Принимает на вход строку в бинарном виде  
# Возвращает двоичный код с подсчитанными контрольными битами  
**def** encode\_shred(binary\_input):  
    power **=** 0  
    control\_byte\_index **=** 2 **\*\*** power **-** 1  
    encoded **=** binary\_input  
    **while** control\_byte\_index < len(encoded):  
        encoded **=** insert\_symbol(encoded, '-', control\_byte\_index)  
        power **+=** 1  
        control\_byte\_index **=** 2 **\*\*** power **-** 1  
    power **=** **-**1  
    **for** k **in** range(len(encoded)):  
        **if** encoded[k] **==** '-':  
            power **+=** 1  
            byte\_value **=** 0  
            cur\_index **=** k  
            **while** cur\_index <**=** len(encoded):  
                start **=** cur\_index  
                end **=** cur\_index **+** (2 **\*\*** power)  
                **for** i **in** range(start, end):  
                    cur\_index **+=** 1  
                    **if** i >**=** len(encoded):  
                        **break**  
                    **if** encoded[i] **==** '-':  
                        **continue**  
                    **else**:  
                        byte\_value **+=** int(encoded[i])  
                cur\_index **+=** (2 **\*\*** power)  
            encoded **=** encoded.replace('-', str(byte\_value **%** 2), 1)  
    **return** encoded  
    
# decode\_shred Отобразить информационные биты  
# Принимает на вход двоичный код Хэмминга  
# Возвращает 2 переменных,  
# 1. Исходное сообщение с исправленной ошибкой,  
# 2. Сообщение с кодом Хэмминга,  
# 3. Бит с ошибкой (если нет, то = 0)  
**def** decode\_shred(binary\_input):  
    power **=** 0  
    control\_bytes **=** []  
    **while** (2 **\*\*** power **-** 1) < len(binary\_input):  
        byte\_value **=** 0  
        cur\_index **=** (2 **\*\*** power **-** 1)  
        **while** cur\_index <**=** len(binary\_input):  
            start **=** cur\_index  
            end **=** cur\_index **+** (2 **\*\*** power)  
            **for** i **in** range(start, end):  
                cur\_index **+=** 1  
                **if** i >**=** len(binary\_input):  
                    **break**  
                **if** i **+** 1 **==** 2 **\*\*** power:  
                    **continue**  
                byte\_value **+=** int(binary\_input[i])  
            cur\_index **+=** (2 **\*\*** power)  
        control\_bytes.append(byte\_value **%** 2)  
        power **+=** 1  
    
    power **=** 0  
    error\_byte\_n **=** 0  
    i **=** 0  
    **while** (2 **\*\*** power **-** 1) < len(binary\_input):  
        **if** int(binary\_input[2 **\*\*** power **-** 1]) !**=** control\_bytes[i]:  
            error\_byte\_n **+=** 2 **\*\*** power  
        i **+=** 1  
        power **+=** 1  
    ch\_binary\_input **=** binary\_input  
    **if** error\_byte\_n !**=** 0:  
        ch\_binary\_input **=** replace\_by\_index(binary\_input, str(abs(  
            int(binary\_input[error\_byte\_n **-** 1]) **-** 1)), error\_byte\_n **-** 1)  
    message **=** ''  
    power **=** 0  
    **for** i **in** range(len(ch\_binary\_input)):  
        **if** i **+** 1 !**=** 2 **\*\*** power:  
            message **+=** ch\_binary\_input[i]  
        **else**:  
            power **+=** 1  
    **return** message, binary\_input, error\_byte\_n  
    
d **=** ''  
**while** **not** (d **in** ['1', '2', '3', '4']):  
    d **=** input('''Введите число от 1 до 2  
    1. Закодировать сообщение в код Хэмминга  
    2. Раскодировать сообщение из кода Хэмминга  
    -> ''')  
    **if** d **==** '1':  
        **try**:  
            i **=** input('Введите двоичный код -> ')  
            int(i, 2)  
            print("Сообщение в коде Хэмминга:", encode\_shred(i))  
        **except**:  
            print('Неверный формат ввода данных')  
    **elif** d **==** '2':  
        **try**:  
            i **=** input('Введите двоичный код -> ')  
            int(i, 2)  
            o1, o2, o3 **=** decode\_shred(i)  
            print()  
            print('Исходное сообщение с исправленной ошибкой:',o1)  
            print('Сообщение с кодом Хэмминга:', o2)  
            print('Бит с ошибкой (если нет, то = 0):', o3)  
        **except**:  
            print('Неверный формат ввода данных')

Решение заданий 1-2 через программу:

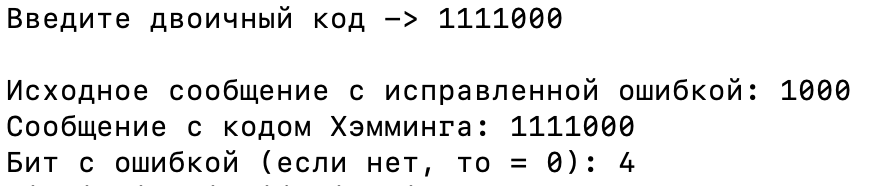
Код 52, задание 1:



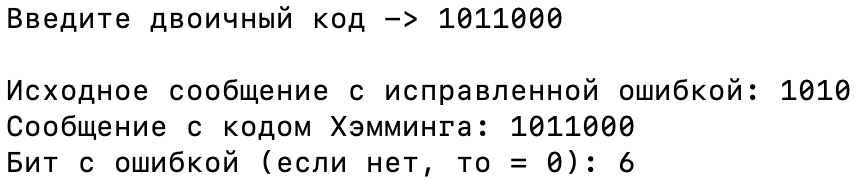
Код 89, задание 1:



Код 14, задание 1:



Код 11, задание 1:



Код 20, задание 2:

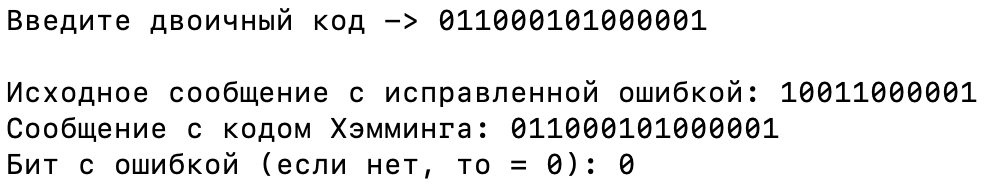
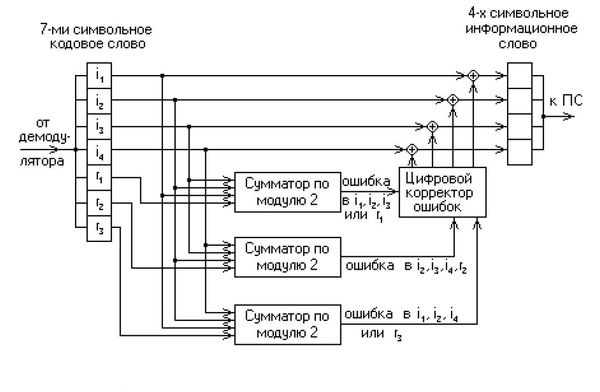


Схема декодирования кода Хэмминга



Задание 8

(52+89+14+11+20)\*4=744

2 в степени 9 < 744 < 2 в степени 10

Следовательно, будет 9 проверочных разрядов.

9 / (744 + 9) = 0.012 – коэффициент избыточности.

# Вывод

Во время выполнения лабораторной работы я изучил работу самовосстанавливающегося кода Хэмминга, смог реализовать кодирование и декодирование программно и выполнить все задания. Полученные знания пригодятся мне в будущем.