Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Пермский национальный исследовательский

политехнический университет»

(ПНИПУ)

Кафедра вычислительной математики и механики

Отчёт

по дисциплине: «Технологии обработки информации»

по теме: «Код Хэмминга»

Выполнил:

Студент группы ИСТ-19-1бзу

Соболь Евгений Валерьевич

Проверил:

доцент кафедры ВМиМ

Сабатулина Татьяна Леонидовна

Пермь

2021 г.

**Листинг программы с комментариями по каждой функции:**

from math import log2  
  
  
def add\_check\_zero(line: str) -> str:  
 *"""Функция добавляет контрольные биты"""* for i in range(int(log2(len(line)))+1):  
 line = line[:2 \*\* i - 1] + "0" + line[2 \*\* i - 1:]  
 return line  
  
  
def create\_line(length: int, k: int) -> str:  
 *"""Функция создаёт строку, начиная с n значения, чередует X и O по n раз"""* s = ""  
 flag = 0  
 for i in range(k-1):  
 s += "O"  
 while len(s) < length:  
 for i in range(k):  
 if len(s) < length:  
 if flag % 2 == 0:  
 s += "X"  
 else:  
 s += "O"  
 else:  
 break  
 flag += 1  
 return s  
  
  
def create\_matrix(m: list) -> list:  
 *"""Функция создаёт список из строк, который будет использоваться как матрица"""* for i in range(int(log2(len(m[0])))+1):  
 m.append(create\_line(len(m[0]), 2\*\*i))  
 return m  
  
  
def output\_matrix(m: list):  
 *"""Вспомогательная функция для вывода матрицы"""* for i in m:  
 print(i)  
 print()  
  
  
def check\_value\_control(matrix: list) -> str:  
 *"""Функция сравнивает значения в матрице и проставляет контрольные биты"""* sum\_one = {}  
 for i in range(int(log2(len(matrix[0])))+1):  
 sum\_one[2\*\*i] = 0  
 for i in range(1, len(matrix)):  
 for j in range(len(matrix[i])):  
 if matrix[i][j] == "X" and matrix[0][j] == "1":  
 sum\_one[2\*\*(i-1)] = sum\_one[2\*\*(i-1)] + 1  
 for i in sum\_one.keys():  
 if sum\_one[i] % 2 == 1:  
 matrix[0] = matrix[0][:i-1] + "1" + matrix[0][i:]  
 print("-----ПРОСТАВИМ КОНТРОЛЬНЫЕ БИТЫ>-----")  
 print("|| | | |")  
 print(matrix[0], "\n")  
 return matrix[0]  
  
  
def add\_insign\_zeros(line: str) -> str:  
 *"""Функция дополняет хначения незначимыми контрольными битами 0, до размера 16 бит- 1 символ"""* while len(line) < 16:  
 line = "0" + line  
 return line  
  
  
def encoding(line: str) -> str:  
 *"""Основная функция для кодирования методом Хэмминга"""* print("-----КОДИРОВАНИЕ-----\n")  
 print("ПЕРЕВЕДЁМ СИМВОЛЫ В ACSII КОДЫ, КОДЫ ПЕРЕВЕДЁМ В ДВОИЧНУЮ СИСТЕМУ")  
 print("ДВОИЧНУЮ ЗАПИСЬ ДОПОЛНИМ НЕЗНАЧАЩИМИ НУЛЯМИ до 16-БИТ")  
 line = list(map(lambda x: add\_insign\_zeros("0" + bin(ord(x)).split("b")[1]), list(line)))  
 print("СПИСОК СО ЗНАЧЕНИЯМИ: ", line, "\n")  
 final\_encoding = ""  
 for i in range(len(line)):  
 bit\_16 = line[i]  
 bit\_16 = [add\_check\_zero(bit\_16)]  
 print(f"-({i + 1})-ДОБАВИМ КОНТРОЛЬНЫЕ БИТЫ-----")  
 print("|| | | |")  
 print(bit\_16[0], "\n")  
 mtx\_16bit = create\_matrix(bit\_16)  
 print("-----СОЗДАДИМ МАТРИЦУ-----")  
 output\_matrix(mtx\_16bit)  
 print()  
 final\_encoding += check\_value\_control(mtx\_16bit)  
 return final\_encoding  
  
  
def replace\_with\_null(line: str) -> str:  
 *"""Функция обнуляет контрольные биты"""* for i in range(int(log2(len(line)))+1):  
 line = line[:2\*\*i - 1] + "0" + line[2\*\*i:]  
 return line  
  
  
def delete\_control\_value(line: str) -> str:  
 *"""Функция удаляет контрольные биты"""* for i in range(int(log2(len(line)))+1):  
 line = line[:2\*\*i - (i + 1)] + line[2\*\*i - i:]  
 return line  
  
  
def comparison\_control\_bits(line1: str, line2: str, i\_help: int) -> bool:  
 *"""Функция сравнивает контрольные биты в строках"""* flag = True  
 i\_err = None  
 m\_2i = []  
 for i in range(int(log2(len(line1)))+1):  
 m\_2i.append(i)  
 if line1[2\*\*i-1] != line2[2\*\*i-1]:  
 flag = False  
 print("-----СРАВНИМ КОНТРОЛЬНЫЕ БИТЫ-----")  
 print(line1)  
 print("|| | | |")  
 print(line2)  
 for i in range(len(line2)):  
 flg1 = 0  
 for j in m\_2i:  
 if 2\*\*j == (i+1):  
 flg1 = 1  
 if (flg1 == 0) and (line1[i] != line2[i]):  
 print("------------------>НЕКОРРЕКТНЫЙ БИТ: ", i\_help\*21 + (i+1))  
 print("ИТОГ: ", flag, "\n")  
 return flag  
  
  
def decoding(new\_line: str) -> str:  
 *"""Основная функция декодирования методом Хэмминга, также позволяет исправить одну ошибку на 16 бит информации"""* print("-----ДЕКОДИРОВАНИЕ-----")  
 count\_16bit = len(new\_line) // 21  
 print("Количество символов для декодирования: ", count\_16bit)  
 border = 0  
 final\_decoding = ""  
 for i in range(count\_16bit):  
 line = new\_line[border:border + 21]  
 print(f"-({i + 1})-СИМВОЛ-----")  
 print(line, "\n")  
 border += 21  
 original\_line = replace\_with\_null(line)  
 print(f"ОБНУЛЯЕМ КОНТРОЛЬНЫЕ БИТЫ")  
 print(original\_line, "\n")  
 matrix = create\_matrix([original\_line])  
 print(f"-----СОЗДАЁМ МАТРИЦУ-----")  
 output\_matrix(matrix)  
 original\_line = check\_value\_control(matrix)  
 i\_old = i  
 if comparison\_control\_bits(line, original\_line, i):  
 original\_line = delete\_control\_value(original\_line)  
 print("-----УДАЛЯЕМ КОНТРОЛЬНЫЕ ЗНАЧЕНИЯ-----")  
 print(original\_line, "\n")  
 final\_decoding += chr(int(original\_line, 2))  
 print("-----РАСШИФРОВАННАЯ ЧАСТЬ-----")  
 print(final\_decoding, "\n")  
 else:  
 sum\_err = 0  
 print("------ОБНАРУЖЕНА ОШИБКА-----")  
 for i in range(int(log2(len(original\_line)))+1):  
 print(f"{original\_line[2 \*\* i - 1]} != {line[2 \*\* i - 1]}")  
 if original\_line[2 \*\* i - 1] != line[2 \*\* i - 1]:  
 sum\_err += 2 \*\* i  
 repl = "1" if original\_line[sum\_err - 1] == "0" else "0"  
 original\_line = original\_line[:sum\_err - 1] + repl + original\_line[sum\_err:]  
 original\_line = replace\_with\_null(original\_line)  
 matrix = create\_matrix([original\_line])  
 original\_line = check\_value\_control(matrix)  
 if comparison\_control\_bits(line, original\_line, i\_old):  
 original\_line = delete\_control\_value(original\_line)  
 final\_decoding += chr(int(original\_line, 2))  
 else:  
 print(original\_line)  
 print("Восстановить данные не удалось, больше 2 ошибок на 16 бит информации")  
 exit()  
 return final\_decoding  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 print("Задание № 5 Код Хэмминга")  
 source\_line = input("Введите строку:")  
 encoding\_value = encoding(source\_line)  
 print("--------------------РЕЗУЛЬТАТ--------------------")  
 print("Закодированное значение: ", encoding\_value, "\n")  
 decoding\_value = decoding("000000010000010111001110100001000011000101100000000000011110011010000010000001100001")  
 print("--------------------РЕЗУЛЬТАТ--------------------")  
 print("Расшифрованное значение: ", decoding\_value, "\n")

**Пример работы программы № 1:**

Тестовое значение для ввода: Yes!

**Результат кодирования:**

000000010000010111001110100000000011000101100000000000011110011010000010000001100001

Тестовое значение для декодирования: 000000010000010111001110100000000011000101100000000000011110011010000010000001100001

**Результат декодирования:**

Yes!

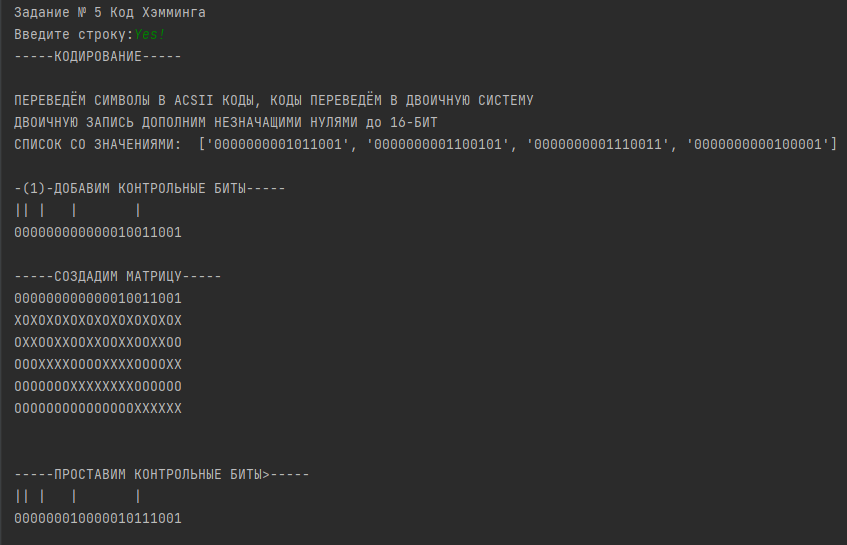


Рисунок - Скриншот программы (кодирование)

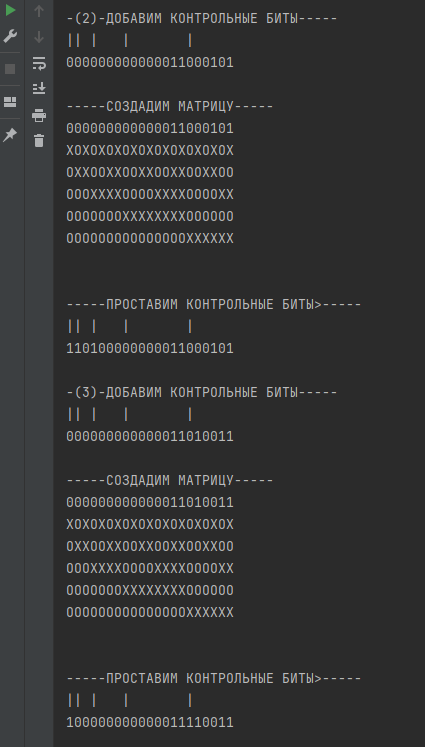


Рисунок - Скриншот программы (кодирование - 2)

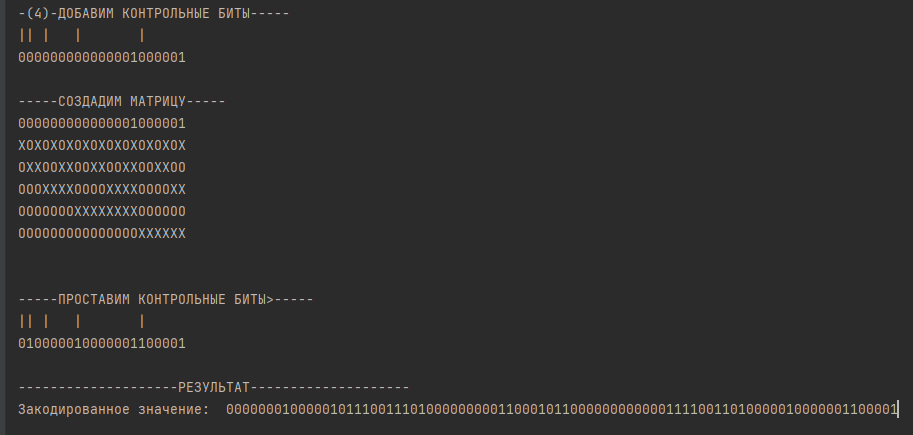


Рисунок - Скриншот программы (кодирование результат)

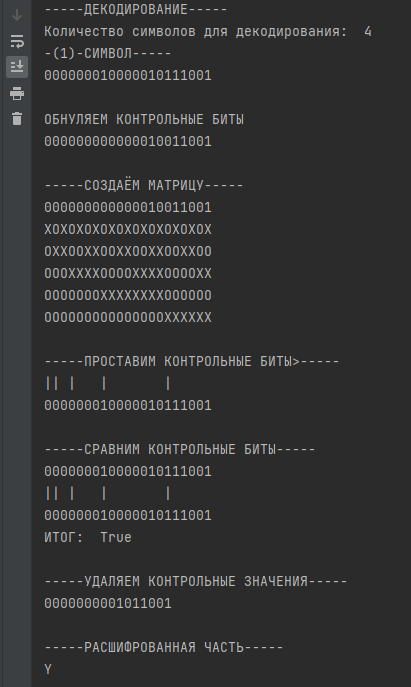


Рисунок –Скриншот программы (декодирование - 1)

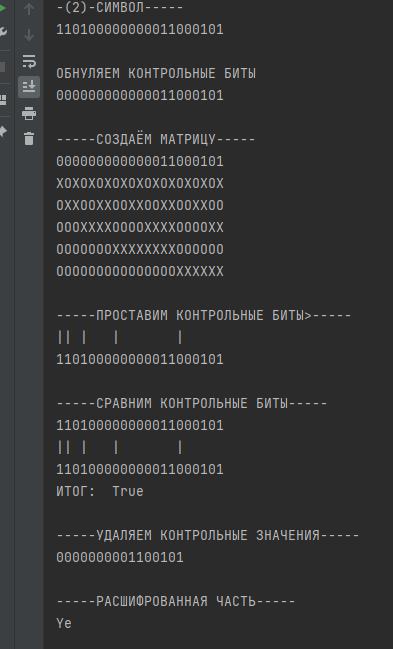


Рисунок –Скриншот программы (декодирование - 2)

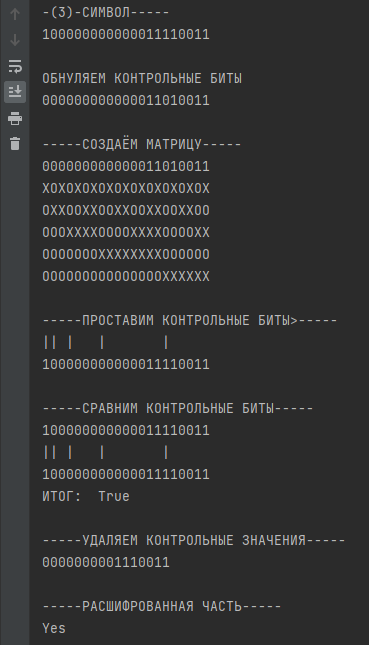


Рисунок –Скриншот программы (декодирование - 3)

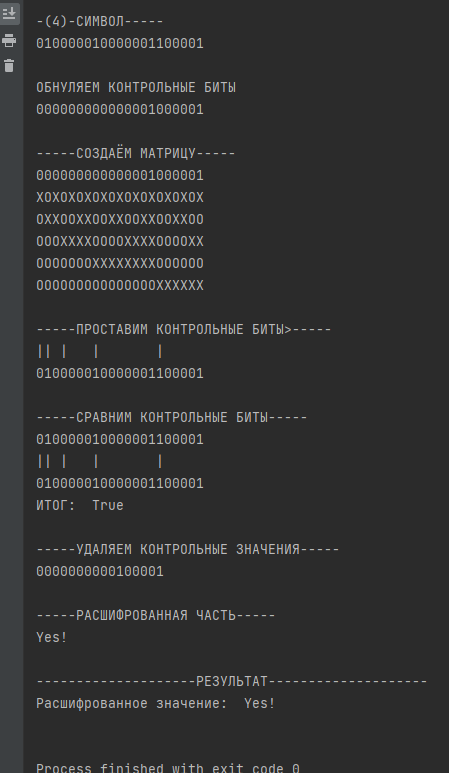


Рисунок –Скриншот программы (декодирование результат)

**Пример работы программы № 2:**

Тестовое значение для ввода: No!

**Результат кодирования:**

110000010000010101110100000000000011001111010000010000001100001

Тестовое значение для декодирования (с ошибкой): 110001010000010101110100000000000011001111010000010000001100001

Бит с ошибкой - 6

**Результат декодирования:**

No!

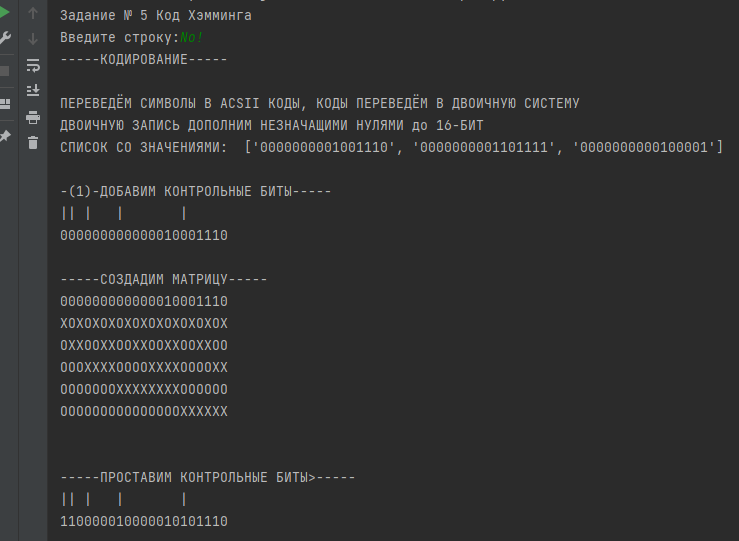


Рисунок –Скриншот программы (кодирование)

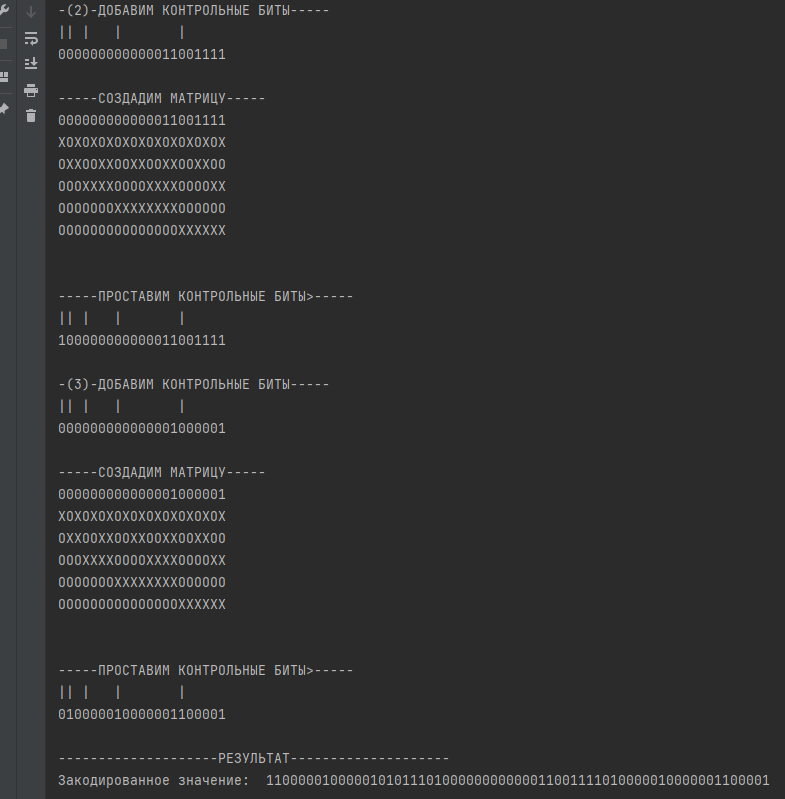


Рисунок - Скриншот программы (кодирование результат)

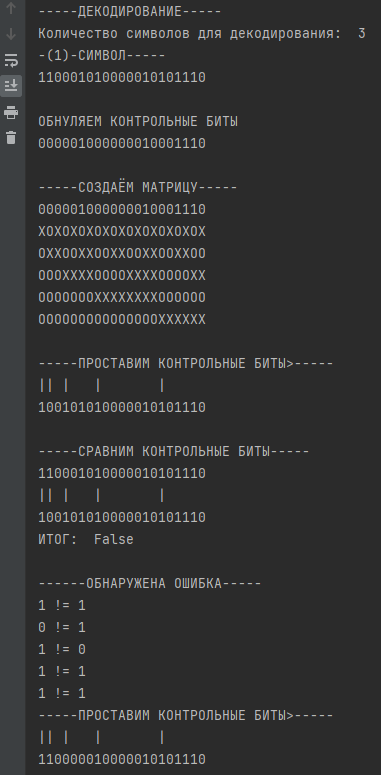


Рисунок - Скриншот программы (декодирование - 1)

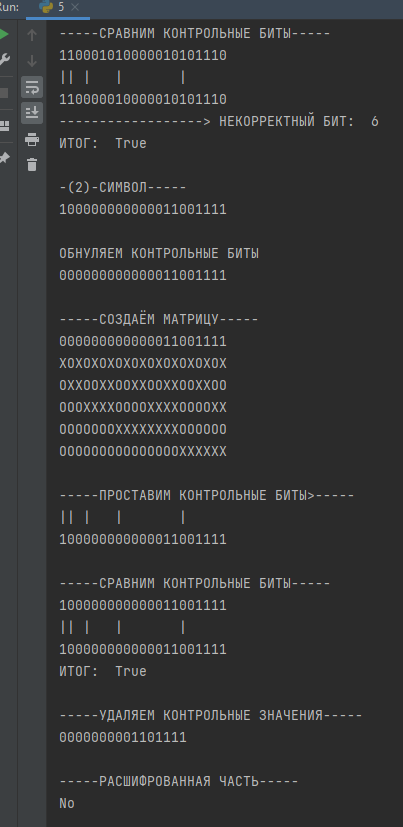


Рисунок - Скриншот программы (декодирование - 2)

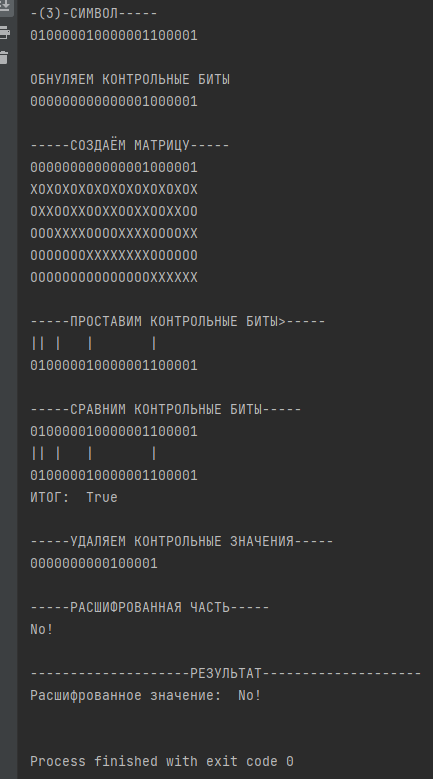


Рисунок - Скриншот программы (декодирование результат)

Тестовое значение для декодирования (с ошибкой): 110000010000010101110100000001000011001111010000010000001100001

Бит с ошибкой - 30

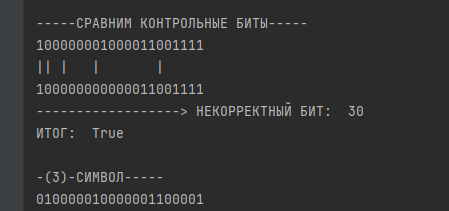


Рисунок - тест отображения ошибочного бита (1)

Тестовое значение для декодирования (с ошибкой): 110000010000010101110100000000000011001111010000010001001100001

Бит с ошибкой - 54

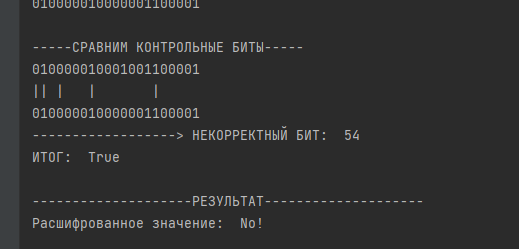


Рисунок – тест отображения ошибочного бита (2)