ВЕСЬ КОД ЗАПУСКАТЬ В ОНЛАЙН КОМПИЛЯТОРЕ

Задание 1

#include <iostream>

#include <bitset>

bool parity\_bit(std::bitset<8> &bits){

    bool parity = 0;

    for(int i=0; i<8; i++)

        parity ^= bits[i];

    return parity;

}

int main(){

    char data;

    while (true){

        std::cout << "Enter the data to be transmitted: ";

        std::cin >> data;

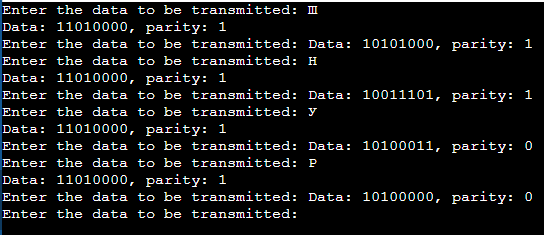
        std::bitset<8> bits(data);

        std::cout << "Data: " << bits << ", parity: " << parity\_bit(bits) << std::endl;

    }

}

Смысл кода, закидываешь букву, она тебе её битовое представление возвращает + паритетный бит.



Задание 2

CRC:

G(x) = x4 + x1 + x0. -> 10011

Ш – 1101 1000

Н – 1100 1101

У – 1101 0011

|  |  |  |  |
| --- | --- | --- | --- |
| Делимое P(x) (входные данные) | 1101 1000 | 1100 1101 | 1101 0011 |
| P(x) \* xN | 1101 1000 0000 | 1100 1101 0000 | 1101 0011 0000 |
| Деление P(x) \* xN mod G(x) |  |  |  |
| Частное | 11001101 | 11011011 | 11000111 |
| Остаток R(x) (контрольная сумма) | 0111 | 1101 | 1001 |

def xor(bitset1, bitset2):

return [bitset1[i] ^ bitset2[i] for i in range(len(bitset1))]

def get\_sum(bitset):

bitset += [0,0,0,0]

temp = bitset[0:5]

chastnoe = []

for i in range(5, len(bitset), 1):

if temp[0] == 1:

temp = xor(temp, [1,0,0,1,1])

chastnoe.append(1)

else:

temp = xor(temp, [0,0,0,0,0])

chastnoe.append(0)

temp.pop(0)

temp += [bitset[i]]

if temp[0] == 1:

temp = xor(temp, [1,0,0,1,1])

chastnoe.append(1)

else:

temp = xor(temp, [0,0,0,0,0])

chastnoe.append(0)

temp.pop(0)

return chastnoe, temp

#11001010

str = "11010000"

print(get\_sum([int(x) for x in str]))

Программа нужна для вычисления частного и проверки остатка R(x).

Задание 3

ЕСС:

А + Н = 11000000110

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Номер позиции бита | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
| Обозначение бита | r1 | r2 | x1 | r3 | x2 | x3 | x4 | r4 | x5 | x6 | x7 | x8 | x9 | x10 | x11 |
| Значение бита, ХR | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 |
| Двоичное представление  номера позиции бита, N | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | r1 | 1 |
| 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | r2 | 0 |
| 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | r3 | 1 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | r4 | 0 |

Проверка целостности:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Номер позиции бита | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
| Обозначение бита | r1 | r2 | x1 | r3 | x2 | x3 | x4 | r4 | x5 | x6 | x7 | x8 | x9 | x10 | x11 |
| Значение бита, ХR’ | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | pb | 0 |
| Двоичное представление  номера позиции бита, N | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | s1 | 0 |
| 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | s2 | 0 |
| 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | s3 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | s4 | 0 |

Ошибок нет. Вектор синдромов состоит из нулей и паритетный бит равен 0.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Номер позиции бита | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
| Обозначение бита | r1 | r2 | x1 | r3 | x2 | x3 | x4 | r4 | x5 | x6 | x7 | x8 | x9 | x10 | x11 |
| Значение бита, ХR’ | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | pb | 1 |
| Двоичное представление  номера позиции бита, N | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | s1 | 1 |
| 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | s2 | 1 |
| 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | s3 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | s4 | 1 |

В последовательности одиночная исправимая ошибка. Для ее исправления находим бит под номером 10112, то есть, 1110 и инвертируем его.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Номер позиции бита | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
| Обозначение бита | r1 | r2 | x1 | r3 | x2 | x3 | x4 | r4 | x5 | x6 | x7 | x8 | x9 | x10 | x11 |
| Значение бита, ХR’ | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | pb | 0 |
| Двоичное представление  номера позиции бита, N | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | s1 | 0 |
| 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | s2 | 0 |
| 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | s3 | 1 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | s4 | 1 |

В последовательности двоичная неисправимая ошибка.

КОД:

ar = [

[1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1],

[0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1],

[0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 1, 1, 1, 1],

[0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1]

]

def get\_sum(bitset: list):

bitset = [0, 0] + [bitset[0]] + [0] + bitset[1:4] + [0] + bitset[4:]

r = [0, 0, 0, 0]

for i in range(4):

for j in range(15):

if ar[i][j] & bitset[j]:

r[i] += 1

return [i % 2 for i in r]

def check\_sum(bitset, sum):

bitset = [sum[0], sum[1]] + [bitset[0]] + [sum[2]] + bitset[1:4] + [sum[3]] + bitset[4:]

r = [0, 0, 0, 0]

pb = 0

for el in bitset:

pb += el

pb %= 2

for i in range(4):

for j in range(15):

if ar[i][j] & bitset[j]:

r[i] += 1

r = [i % 2 == pb for i in r]

if all(r):

return "Сумма верная!", f"{pb=}"

return int("".join([str(int(s)) for s in r[::-1]]), 2), r, f"{pb=}"

str1 = "11001010111"

bitset = [int(x) for x in str1] # input("Введите последовательность: "`)

sum = get\_sum(bitset)

print(sum)

#bitset[6] = not bitset[6]

#bitset[3] = not bitset[3]

print(check\_sum(bitset, sum))

По заданию нужно допустить 2 ошибки, это делается 2-мя строками bitset[3] = not bitset[3] и bitset[6] = not bitset[6]. Их просто нужно раскоментарить - убрать #.