**Московский Государственный технический университет**

**имени Н. Э. Баумана**



Домашнее задание по курсу:

*«Сети и телекоммуникации*»

Вариант 6.

Работу выполнил студент группы ИУ5-53

\_\_\_\_\_\_\_\_\_\_\_\_Кондратьев Максим

Работу проверил:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Галкин В.А.

Москва, 2018

**Содержание:**

Оглавление

Цель. 3

Постановка задачи. 3

Алгоритм решения. 3

Алгоритм декодирования. 4

Алгоритм работы программы 5

Результирующая таблица. 7

Приложение №1 8

Исходные тексты программы. 8

Приложение №2 10

Результат работы программы 10

Выводы. 11

Список используемой литературы. 12

### Цель.

Целью домашнего задания является приобретение и закрепление студентами практических навыков по разработке и реализации алгоритмов кодирования и декодирования корректирующим кодом, а также определение реальной обнаруживающей или корректирующей способности этого кода.

### Постановка задачи.

Имеется дискретный канал связи, на вход которого подается закодированная в соответствии с вариантом задания кодовая последовательность. В канале возможны ошибки любой кратности. Вектор ошибки может принимать значения от единицы в младшем разряде до единицы во всех разрядах кодового вектора. Для каждого значения вектора ошибки на выходе канала после декодирования определяется факт наличия ошибки и предпринимается попытка ее исправления.

Обнаруживающая способность кода Cо определяется как отношение числа обнаруженных ошибок No к общему числу ошибок данной кратности, которое определяется как число сочетаний из n (длина кодовой комбинации) по i (кратность ошибки – число единиц в векторе ошибок) - Cin.

Cо = No / Cin

Необходимо определить корректирующую способность кода. Исходные данные:

|  |  |  |  |
| --- | --- | --- | --- |
| № варианта | Информационный  вектор | Код | Способность  кода |
| 6 | 1110 | X [7, 4] | Co |

### Алгоритм решения.

Пусть информационный кодовый вектор v = 1110 (4 разряда значащих, 7 всего). В коде Хэмминга этот вектор будет занимать позиции c3, c5, c6, c7, начиная с младшего разряда, а позиции c1, c2, c4 отводятся под проверочные разряды кода. Пронумеруем все позиции кода в двоичной системе счисления: c111 c110 c101 [c100] c011 [c010] [c001] и выделим позиции для размещения проверочных разрядов. Определим значения проверочных разрядов кода суммированием по mod2 тех разрядов кода, в номере которых двоичный разряд с (i)-ым весом равен единице.

v = 1110

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| c111 | c110 | c101 | c100 | c011 | c010 | c001 |
| 1 | 1 | 1 | 1 | 0 | 0 | 0 |

Значения c100, c010, c001 высчитаем как:

Таким образом, получен кодовый вектор v' = 1111000, который передается по каналу, подверженному влиянию помех. Пусть вектор ошибки равен e=0000100, тогда принятая из дискретного канала кодовая комбинация будет иметь вид:

v'' = 1111100 = v' e.



### Алгоритм декодирования.

Вычисляется значение синдрома ошибки:

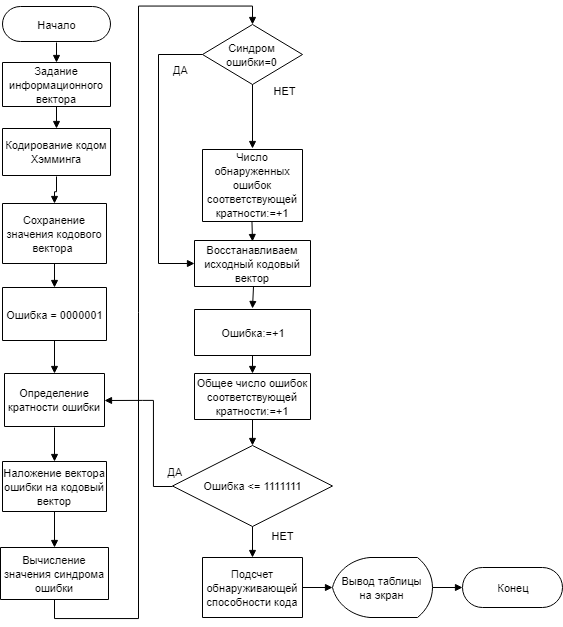
Eош = || hrhr-1 ...hi...h1 ||.

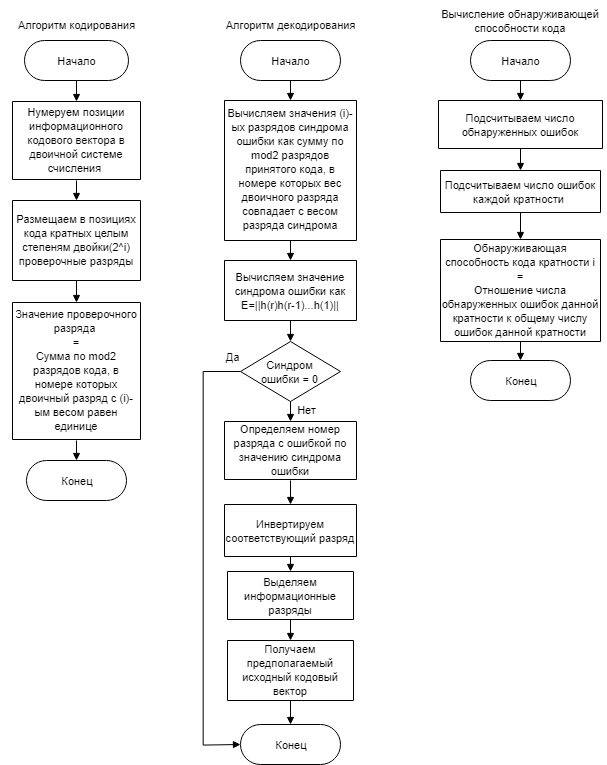
Значение (i)-го разряда синдрома определяется как сумма по mod2 тех разрядов принятого кода, включая проверочные, в номере которых вес двоичного разряда совпадает с весом разряда синдрома. Для нашего примера v'' = 1111100

Eош = || h3h2h1 || = ||011|| - синдром ошибки определяет в двоичной системе номер разряда, в котором обнаружена однократная ошибка.

Для исправления ошибки необходимо инвертировать третий разряд - c3. Получим: v'' = 1111000, откуда, выделяя информационные разряды, получаем исходный кодовый вектор v = 1110.

### Алгоритм работы программы





### Результирующая таблица.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  | Примечание |
| 1 | 7 | 7 | 1 | 7 | 1 | Исправляется любая ошибка кратности 1 |
| 2 | 21 | 21 | 1 | 0 | 0 | Ошибки кратности 2 и более не корректируются |
| 3 | 35 | 28 | 0,8 | 0 | 0 | Наблюдается симметрия обнаруживающей способности относительно средней кратности |
| 4 | 35 | 28 | 0,8 | 0 | 0 | Наблюдается симметрия обнаруживающей способности относительно средней кратности |
| 5 | 21 | 21 | 1 | 0 | 0 |  |
| 6 | 7 | 7 | 1 | 0 | 0 |  |
| 7 | 1 | 0 | 0 | 0 | 0 | Ошибка максимальной кратности не обнаруживается |

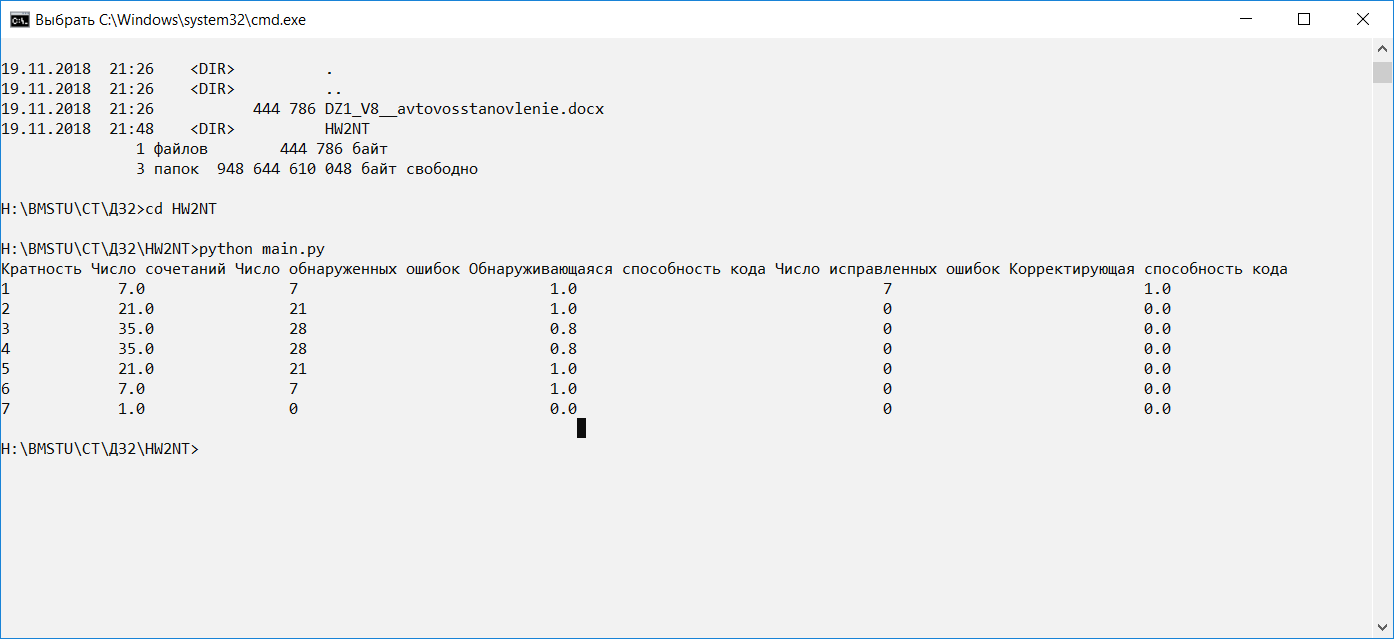
### Приложение №1

### Исходные тексты программы.

from math import factorial  
  
  
def number\_to\_bv(number, depth=7):  
 vector = [0]\*depth  
 counter = depth  
 while number:  
 counter -= 1  
 res = number % 2  
 number //= 2  
 vector[counter] = res  
 return vector  
  
  
def bv\_to\_number(bit\_v):  
 power = 0  
 number = 0  
 for bit in bit\_v[::-1]:  
 number += bit\*pow(2, power)  
 power += 1  
 return number  
  
  
def combination(n, k):  
 return factorial(n)/(factorial(k)\*factorial(n-k))  
  
  
def multiplicity(vector):  
 m = 0  
 for i in vector:  
 if i == 1:  
 m += 1  
 return m  
  
  
def err\_synd(v):  
 b\_v = v[::-1]  
 err\_s = [0]\*3  
 err\_s[2] = (b\_v[0] ^ b\_v[2] ^ b\_v[4] ^ b\_v[6])  
 err\_s[1] = (b\_v[1] ^ b\_v[2] ^ b\_v[5] ^ b\_v[6])  
 err\_s[0] = (b\_v[3] ^ b\_v[4] ^ b\_v[5] ^ b\_v[6])  
 return err\_s  
  
  
def err\_fix(v, err\_s):  
 f\_err\_v = [0] \* len(v)  
 n\_b = bv\_to\_number(err\_s)  
 if n\_b != 0:  
 f\_err\_v[n\_b - 1] = 1  
 fixed\_v = my\_xor(v, f\_err\_v[::-1])  
 return fixed\_v  
  
  
def my\_xor(iv, ev):  
 return list(map(lambda x: x[0] ^ x[1], zip(iv, ev)))  
  
  
def print\_table(table):  
 for header in table[0].keys():  
 print(header, end=' ')  
 print()  
 for table\_row in table:  
 for res in table\_row.items():  
 str\_l = len(res[0]) - len(res[1].\_\_str\_\_()) + 4  
 print(res[1], end=' ' \* str\_l)  
 print()  
  
  
err = 0  
inf\_v = [1, 1, 1, 1, 0, 0, 0]  
table = [{'Кратность': i,  
 'Число сочетаний': combination(7, i),  
 'Число обнаруженных ошибок': 0,  
 'Обнаруживающаяся способность кода': 0,  
 'Число исправленных ошибок': 0,  
 'Корректирующая способность кода': 0} for i in range(1, 8)]  
while True:  
 err += 1  
  
 err\_v = number\_to\_bv(err)  
 m = multiplicity(err\_v)  
 e\_inf\_v = my\_xor(inf\_v, err\_v)  
 e\_s = err\_synd(e\_inf\_v)  
  
 if any(e\_s):  
 table[m-1]['Число обнаруженных ошибок'] += 1  
  
 f\_inf\_v = err\_fix(e\_inf\_v, e\_s)  
  
 if inf\_v == f\_inf\_v:  
 table[m - 1]['Число исправленных ошибок'] += 1  
  
 if all(number\_to\_bv(err)):  
 break  
  
for t\_r in table:  
 t\_r['Обнаруживающаяся способность кода'] = t\_r['Число обнаруженных ошибок'] / t\_r['Число сочетаний']  
 t\_r['Корректирующая способность кода'] = t\_r['Число исправленных ошибок'] / t\_r['Число сочетаний']  
  
print\_table(table)

### Приложение №2

### Результат работы программы



### Выводы.

1) Код Хемминга обладает хорошей обнаруживающей способностью. При небольших (1 и 2) и больших (5 и 6) кратностях вектора ошибки обнаруживающая способность равно 100%.

2) Обнаруживающая способность меняется с изменением кратности ошибки на 2 и меняется симметрично относительно ошибки средней кратности (т.е. относительно кратности 3 и 4 в случае 7-ми разрядных векторов).

3) Ошибку наибольшей кратности код Хэмминга не обнаруживает.

4) Код Хэмминга позволяет исправить любую ошибку кратности 1.

5) Использование кода Хэмминга для корректировки при кратности ошибок более 1 нежелательно, ввиду его неспособности исправлять данные ошибки.

### Список используемой литературы.

1. Галкин В.А., Григорьев Ю.А. Телекоммуникации и сети: Учеб. Пособие для вузов.-М.: Изд-во МГТУ им.Н.Э.Баумана, 2003