|  |  |
| --- | --- |
|  | **Министерство науки и высшего образования Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |

|  |  |
| --- | --- |
| ФАКУЛЬТЕТ | РАДИОТЕХНИЧЕСКИЙ |

|  |  |
| --- | --- |
| КАФЕДРА | СИСТЕМЫ ОБРАБОТКИ ИНФОРМАЦИИ И УПРАВЛЕНИЯ |

**ОТЧЕТ ПО ДОМАШНЕМУ ЗАДАНИЮ**

Студент Столярова Ольга Денисовна .

*фамилия, имя, отчество*

Группа РТ5-51Б .

Название предприятия МГТУ им. Н. Э. Баумана ю

Студент Столярова О.Д.

Преподаватель Галкин В.А.

*2021 г.*

***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

В каждом варианте задания необходимо определить либо обнаруживающую, либо корректирующую способность кода. Результаты работы программы представить в виде таблицы:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| i | Cin | No или Nk | Cо или Ck | Примечание |
|  |  |  |  |  |

***Вариант 18***

**

Информационный вектор: 1000

Кодирование циклическим кодом

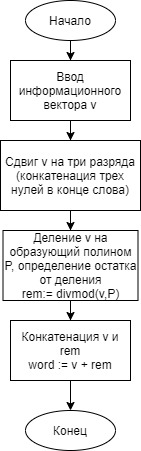
Необходимо определить обнаруживающую способность кода

**Алгоритм кодирования**

Получен информационный вектор v. Осуществляется сдвиг этого вектора на три разряда, т.е. в конце вектора дописываются три нуля.

Полученное значение делится на образующий полином P – в случае кода [7,4] это полином P = x3 + x + 1 (1011). Деление осуществляется как обычное деление в столбик, только производится не вычитание, а сложение по модулю два.

Остаток от деления rem дописывается к изначально заданному вектору v (конкатенация v и rem).



*Пример кодирования*

Информационный вектор: 1000

1. Сдвигаем вектора влево на три разряда, получаем 1000.000
2. Делим полученное кодовое слово на образующий полином 1011, остаток равен – 101
3. Дописываем полученный остаток к изначальному вектору. Получаем слово, закодированное циклическим кодом: 1000.101

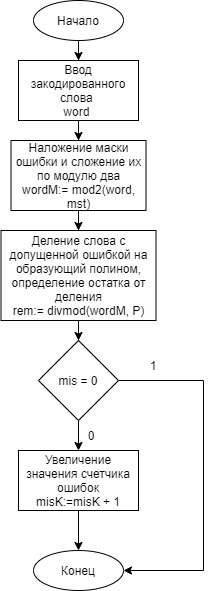
**Алгоритм декодирования**

Получено закодированное слово word. В этом слове допускается ошибка определенной кратности в определенном байте.

Слово с ошибкой wordM делится на образующий полином P – в случае кода [7,4] это полином P = x3 + x + 1 (1011). Деление осуществляется как обычное деление в столбик, только производится не вычитание, а сложение по модулю два.

Если остаток от деления равен нулю, значит ошибки нет.

Если остаток от деления не равен нулю, следовательно в слове обнаружена ошибка – счетчик ошибок увеличивается на единицу misK:= misK + 1.



*Пример декодирования*

* Пусть в слове будет допущена ошибка в 5 разряде: вектор ошибки равен e(x) = x4.

Слово с ошибкой: 1010.000

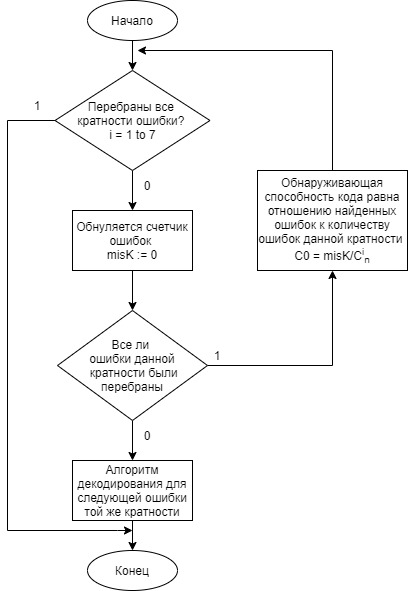
1. Поделим полученный вектор на образующий полином 1011. Остаток от деления: 110.
2. Остаток не равен нулю, следовательно, в слове допущена ошибка.

* Пусть вектор ошибки равен e(x) = x4 + x2 + x – ошибка кратности 3

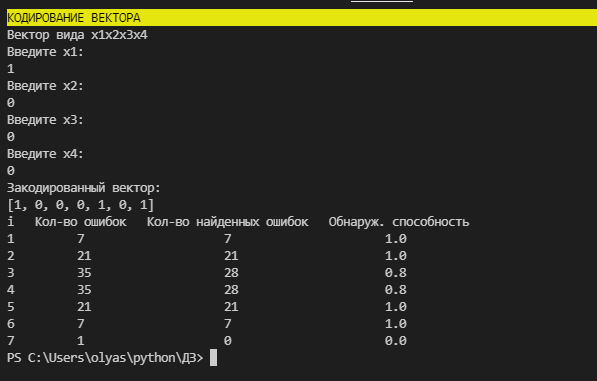
Слово с ошибкой: 1000.110

Если поделить полученный вектор на образующий полином, то остаток будет равен нулю – это пример, показывающий, что обнаруживающая способность кода при ошибке с кратностью больше единицы не стопроцентная.

Алгоритм декодирования происходит в цикле перебора всех ошибок одной кратности, который находится в цикле перебора ошибок всех кратностей.



**Результат выполнения программы:**

****

**Код программы**

Алгоритм выполнен на языке python

**Основной файл - main.py – выполнение функция кодирования, декодирования, наложения маски и вывод таблицы**

from pack import \*

from pack.miscode import \*

from pack.decode import \*

from pack.codir import \*

from colorama import init

init()

from colorama import Fore, Back, Style

mist = []

vect = []

print("Закодированный вектор:")

#НАЛОЖЕНИЕ МАСКИ ОШИБКИ

def mask(mist, vect):

    vect1 = vect.copy()

    t = 0

    while t < 7:

        if mist[t] == 1:

            vect1[t] = xor(vect[t], 1)

        t += 1

    return vect1

def prov(vect):

    v = dely(vect)

    t = 0

    while t < 4:

        if v[t] == 1:

            t = 20

        t += 1

    if t == 21:

        return 1

    else:

        return 0

codvec = code(vec)

print(codvec)

#codvec1 = ['','','','','','','']

#i = 0

#while i < 7:

#    codvec1[i] = codvec[i]

#    i += 1

codvec1 = codvec.copy()

print("i   Кол-во ошибок   Кол-во найденных ошибок   Обнаруж. способность")

l = 0

kol = 0

while l < 7:

     codvec2 = codvec1

     a = mask(l1[l],codvec2)

     prov(a)

     if prov(a) == 1:

         kol += 1

     l += 1

print("1","       ",len(l1),'                  ',kol,'                    ',kol/len(l1))

l = 0

kol = 0

while l < 21:

     codvec2 = codvec1

     a = mask(l2[l],codvec2)

     prov(a)

     if prov(a) == 1:

         kol += 1

     l += 1

print("2","       ",len(l2),'                 ',kol,'                   ',kol/len(l2))

l = 0

kol = 0

while l < 35:

     codvec2 = codvec1

     a = mask(l3[l],codvec2)

     prov(a)

     if prov(a) == 1:

         kol += 1

     l += 1

print("3","       ",len(l3),'                 ',kol,'                   ',kol/len(l3))

l = 0

kol = 0

while l < 35:

     codvec2 = codvec1

     a = mask(l4[l],codvec2)

     prov(a)

     if prov(a) == 1:

         kol += 1

     l += 1

print("4","       ",len(l4),'                 ',kol,'                   ',kol/len(l4))

l = 0

kol = 0

while l < 21:

     codvec2 = codvec1

     a = mask(l5[l],codvec2)

     prov(a)

     if prov(a) == 1:

         kol += 1

     l += 1

print("5","       ",len(l5),'                 ',kol,'                   ',kol/len(l5))

l = 0

kol = 0

while l < 7:

     codvec2 = codvec1

     a = mask(l6[l],codvec2)

     prov(a)

     if prov(a) == 1:

         kol += 1

     l += 1

print("6","       ",len(l6),'                  ',kol,'                    ',kol/len(l6))

l = 0

kol = 0

while l < 1:

     codvec2 = codvec1

     a = mask(l7[l],codvec2)

     prov(a)

     if prov(a) == 1:

         kol += 1

     l += 1

print("7","       ",len(l7),'                  ',kol,'                    ',kol/len(l7))

**Файл decode.py – алгоритм декодирования**

delim = []

delit = [1,0,1,1]

def xor(a,b):

    if a == b:

        return 0

    if a != b:

        return 1

def gxor(delim):

    ost= [0,0,0,0]

    i = 3

    while i >= 0:

         ost[i] = xor(delim[i],delit[i])

         i = i - 1

    return(ost)

def dely(delim):

    delim1 = [0,0,0,0]

    otvet = []

    i = 0

    while i < 4:

        delim1[i] = delim[i]

        i = i + 1

    k = 0

    i = 3

    c = 0

    while i < 7 and k < 3:

         if delim1[0] == 0:

             i = i + 1

             delim1[0] = delim1[1]

             delim1[1] = delim1[2]

             delim1[2] = delim1[3]

             delim1[3] = delim[i]

             k = k + 1

             c = c + 1

             if c == 2:

                  otvet.append(0)

                  c = 0

         if delim1[0] != 0:

             delim1 = gxor(delim1)

             otvet.append(1)

    if k == 2:

        delim1[0] = delim1[1]

        delim1[1] = delim1[2]

        delim1[2] = delim1[3]

        delim1[3] = delim[6]

    return(delim1)

**Файл miscode.py – алгоритм перебора всех вариантов вектора ошибки**

i = 110

k = ''

dvoich = []

dvoichFin = ['','','','','','','']

l1 = []

l2 = []

l3 = []

l4 = []

l5 = []

l6 = []

l7 = []

t = 0

while t < 128:

    b = bin(t)

    for k in b:

       dvoich.append(k)

    i = i + 1

    k = len(dvoich)

    i = k - 1

    j = 0

    while i > 1:

        dvoichFin[j] = dvoich[i]

        i = i - 1

        j = j + 1

    i = 0

    while i < 7:

        if dvoichFin[i] == '':

            dvoichFin[i] = 0

        i = i + 1

    i = 0

    while i < 7:

         dvoichFin[i] = int(dvoichFin[i])

         i = i + 1

    t = t + 1

    kol = 0

    i = 0

    while i < 7:

        if dvoichFin[i] == 1:

            kol = kol + 1

        i = i + 1

    if kol == 1:

        l1.append(dvoichFin)

    if kol == 2:

        l2.append(dvoichFin)

    if kol == 3:

        l3.append(dvoichFin)

    if kol == 4:

        l4.append(dvoichFin)

    if kol == 5:

        l5.append(dvoichFin)

    if kol == 6:

        l6.append(dvoichFin)

    if kol == 7:

        l7.append(dvoichFin)

    dvoichFin = ['','','','','','','']

    dvoich = []

    kol = 0

**codir.py – алгоритм кодирования**

from pack.decode import \*

from colorama import init

init()

from colorama import Fore, Back, Style

vec = [0,0,0,0,0,0,0]

print(Back.YELLOW + Fore.BLACK + 'КОДИРОВАНИЕ ВЕКТОРА'+ Style.RESET\_ALL)

print("Вектор вида x1x2x3x4")

print("Введите x1:")

vec[0] = int(input())

print("Введите x2:")

vec[1] = int(input())

print("Введите x3:")

vec[2] = int(input())

print("Введите x4:")

vec[3] = int(input())

def code(vec):

     vec1 = dely(vec)

     vec[4] = vec1[1]

     vec[5] = vec1[2]

     vec[6] = vec1[3]

     return vec

**Список литературы**

1. Галкин В.А., Григорьев Ю.А. Телекоммуникации и сети: учеб. Пособие для вузов. – М.: Издательство МГТУ им. Н.Э. Баумана, 2003. 608с.
2. М.Н.Аршинов, Л.Е.Садовский. КОДЫ И МАТЕМАТИКА М.: Наука, 1983. 144с.
3. Вернер М. Основы кодирования. – М.: Техносфера, 2004, 288с.
4. Питерсон У., Уэлдон Э. Коды, исправляющие ошибки. М.: Мир", 1976. 593 с.

**Ссылка на GitHub:**

https://github.com/OlyaSto/Olyabmstu