МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ

ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет информационных технологий

Кафедра информационных систем и технологий

Специальность Информационные системы и технологии

**ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №5 НА ТЕМУ:**

**Избыточное кодирование данных в информационных системах. Итеративные коды**

Выполнила студентка 3 курса 1 группы

Пригодич Вера Валерьевна

Минск 2022

**Задание 1:** Разработать собственное приложение, которое позволяет выполнять следующие операции: вписывать произвольное двоичное представление информационного слова *Xk* (кодируемой информации) длиной *k* битов в двумерную матрицу размерностью в соответствии с вариантом либо в трехмерную матрицу в соответствии с вариантом

Для выполнения задания был использован следующий код на языке *python*:

import numpy as np

#произвольное двоичное слово длиной n

def rand\_bin(n):

    bin\_str=[]

    for i in range(n):

       temp = random.randint(0, 1)

       bin\_str.append(temp)

    return bin\_str

# произовльное двоичное представление информционного слова

xk = rand\_bin(k)

# вписать призвольное Xk в двумерную матрицу

k1 = 4

k2 = 6

matrix = np.reshape(xk, (k1, k2))

# вписать произвольное Xk в трехмерную матрицу

k1 = 6

k2 = 2

z = 2

matrix = np.reshape(xk, (z, k1, k2))

Листинг 1 – задание 1

**Задание 2:** вычислять проверочные биты (биты паритетов): а) по двум; б) по трем; в) по четырем направлениям (группам паритетов).

Для выполнения этого задания код, разработанный в предыдущем задании был дополнен следующим кодом:

# вычислять проверочные биты по группам

def mod2(x: int):

    return x % 2

def check\_bits(matrix, groups, k1, k2):

    xr=[]

    for i in range (0,2):

        xr\_temp = np.sum(matrix, axis=i)

        xr = np.append(xr, mod2(xr\_temp)).astype(int)

    if (groups >= 3):

        for i in range (int(-k1/2 - 1), k2):

            xr\_temp = np.sum(np.diag(matrix, i))

            xr = np.append(xr, mod2(xr\_temp))

    if (groups == 4):

        for i in range (int(-k1/2 - 1), k2):

            xr\_temp = np.sum(np.diag(np.fliplr(matrix), i))

            xr = np.append(xr, mod2(xr\_temp))

    return xr

Листинг 2 – задание 2

**Задание 3:** Формировать кодовое слово *Xn* присоединением избыточных символов к информационному слову.

Дополним разработанный в предыдущем задании код следующим образом:

# формировать кодовое слово Xn

xn = np.hstack([xk, xr])

Листинг 3 – задание 3

**Задание 4:** Генерировать ошибку произвольной кратности (*i, i* > 0), распределенную случайным образом среди символов слова *Xn*, в результате чего формируется кодовое слово *Yn.*

Дополним разработанный в предыдущем задании код следующей функцией:

# генерировать ошибку произвольной кратности

def rand\_mistake(matrix, k1, k2, amount = 1):

    yk = matrix

    for n in range (amount):

        i = random.randint(0, k1-1)

        j = random.randint(0, k2-1)

        if (yk[i,j] == 0):

            yk[i,j] = 1

        else:

            yk[i,j] = 0

    return yk

Листинг 4 – задание 4

**Задание 5:** Определять местоположение ошибочных символов итеративным кодом в слове *Yn* в соответствии с используемыми группами паритетов по пункту (2) и исправлять ошибочные символы (результат исправления – слово *Yn’*);

Для выполнения данного задания дополним скрипт следующей функцией:

def find\_mistake(yr, xr, k1, k2, yk):

    n = 0

    i = []

    j = []

    corr = yk

    while n < k2:

        if (yr[n] != xr[n]):

            j.append(n)

        n += 1

    while n < (k1 + k2):

        if (yr[n] != xr[n]):

            i.append(n-k2)

        n += 1

    print("Ошибочные символы по i:",i)

    print("Ошибочные символы по j:",j)

    for elem\_i in i:

        for elem\_j in j:

            if (corr[elem\_i,elem\_j] == 0):

                corr[elem\_i,elem\_j] = 1

            else:

                corr[elem\_i,elem\_j] = 0

    return corr

Листинг 5 – задание 5

**Задание 6:** Выполнять анализ корректирующей способности используемого кода (количественная оценка) путем сравнения соответствующих слов *Xn* и *Yn’*; результат анализа может быть представлен в виде отношения общего числа сгенерированных кодовых слов с ошибками определенной одинаковой кратности (с одной ошибкой, с двумя ошибками и т. д.) к числу кодовых слов, содержащих ошибки этой кратности, которые правильно обнаружены и которые правильно скорректированы.

Дополним разработанный в предыдущем задании код:

words\_number = 50

# Корректирующая способность для 1 ошибки

corr\_number = 0

mistake\_amount = 1

for i in range(words\_number):

    yk = rand\_mistake(matrix, k1, k2, mistake\_amount)

    yr = check\_bits(matrix, groups, k1, k2)

    corr = find\_mistake(yr, xr, k1, k2, yk)

    if (xk == np.reshape(corr, (k1\*k2, ))).all():

        corr\_number += 1

print(f'Корректирующая способность для 1 ошибки: {corr\_number / words\_number}')

# Корректирующая способность для 2 ошибок

mistake\_amount = 2

for i in range(words\_number):

    yk = rand\_mistake(matrix, k1, k2, mistake\_amount)

    yr = check\_bits(matrix, groups, k1, k2)

    corr = find\_mistake(yr, xr, k1, k2, yk)

    if (xk == np.reshape(corr, (k1\*k2, ))).all():

        corr\_number += 1

print(f'Корректирующая способность для 2 ошибок: {corr\_number / words\_number}')

Листинг 5 – задание 6

Результат работы итогового кода:

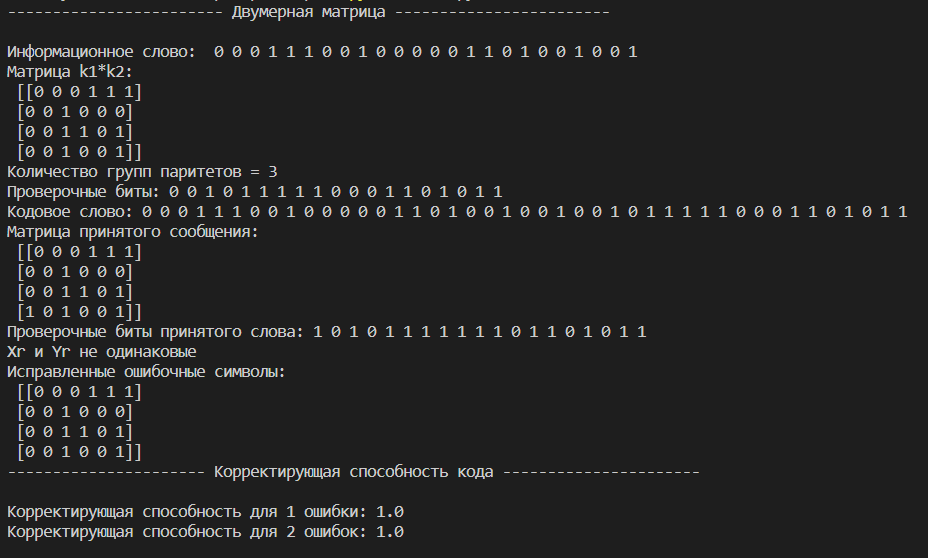


Рисунок 1 ­­­­– Результат работы программы

**Вывод:** в ходе лабораторной работы были приобретены практические навыки кодирования/декодирования двоичных данных при использовании итеративных кодов.