

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ

РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение высшего образования

«Крымский федеральный университет имени В.И. Вернадского»

Физико-технический институт

Кафедра компьютерной инженерии и моделирования

Лабораторная работа № 4

**«Систематический код»**

по дисциплине

«Теория информации и кодирование»

Выполнил:

студент 3 курса

группа ИВТ-222

Гоголев В. Г.

Проверил:

Филиппов Д.М.

«\_\_\_\_» \_\_\_\_\_\_\_20\_\_\_г.

Подпись: \_\_\_\_\_\_\_\_\_\_\_

Симферополь, 2024

**Цель:** построить помехоустойчивый систематический код, позволяющий обнаруживать и исправлять все однократные ошибки.

**Техническое задание:** источник информации вырабатывает сообщения, содержащие k информационных разрядов.

Значения разрядов генерируются в двоичной системе счисления счетчиком случайных чисел. Необходимо: 1. разработать программное обеспечение для передатчика, которое будет строить систематический код с заданной исправляющей способностью; 2. разработать программное обеспечение на приемной стороне, позволяющее корректировать принятую ошибочную кодовую комбинацию; 3. провести комплекс численных экспериментов, в ходе которых на передающей стороне построить систематический код с заданной исправляющей способностью, сгенерировать ошибочный систематический код, на приемной стороне вычислить позицию ошибки и скорректировать принятую кодовую комбинацию.

**Ход работы:**

**Вариант № 4**

**Задание I.**  С использованием разработанного программного обеспечения для передатчика необходимо сгенерировать производящую матрицу систематического кода Pn,k и построить проверочную матрицу Н, при помощи которой можно обнаруживать и исправлять все однократные ошибки.

**Задание II.** Провести цикл комплексных экспериментов (не менее 6), в ходе которого необходимо:

а) сгенерировать случайным образом информационную кодовую комбинацию, состоящую из k разрядов, на передающей стороне;

б) построить для информационной кодовой комбинации на передающей стороне систематический код, позволяющий обнаруживать и исправлять все однократные ошибки;

в) передать проверочную матрицу с выходы программного обеспечения на передающей стороне на вход программного обеспечения приемной стороны;

г) передать систематический код от передатчика к приемнику, сгенерировав случайным образом однократную ошибку в любом разряде систематического кода;

д) при помощи программного обеспечения на приемной стороне определить синдром ошибки принятого систематического кода;

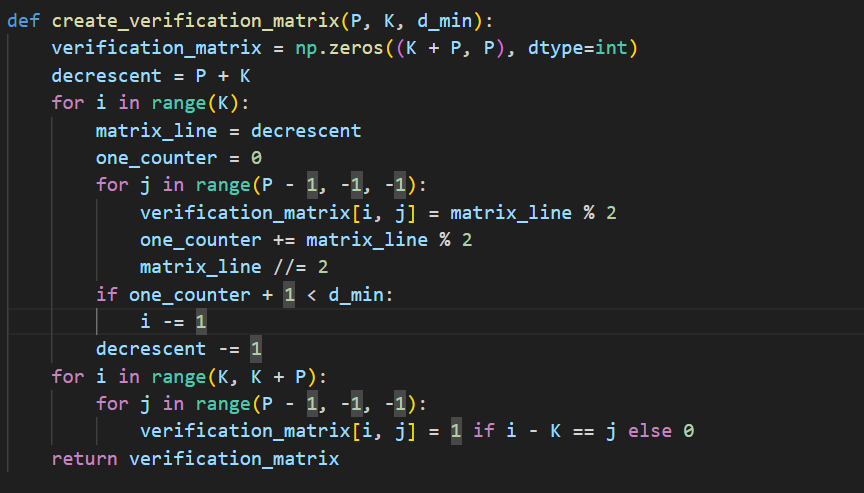
е) с использованием проверочной матрицы на приемной стороне по синдрому ошибки определить позицию ошибки и откорректировать систематический код.

Рисунок 1 – функция creatre\_verification\_matrix

Назначение: создает проверочную матрицу для систематического кода.

Инициализация матрицы: создает нулевую матрицу размером (K + P, P), где K — количество информационных битов, P — количество проверочных битов.

Заполнение строки матрицы:

* Переменная decrescent инициализируется как P + K.
* В цикле строки матрицы заполняются двоичными представлениями чисел в убывающем порядке.
* Подсчитывается количество единиц в строке (one\_counter). Если их недостаточно, строка пересчитывается.
* Добавление единичной матрицы: последние строки матрицы заполняются единицами по диагонали, образуя единичную матрицу.

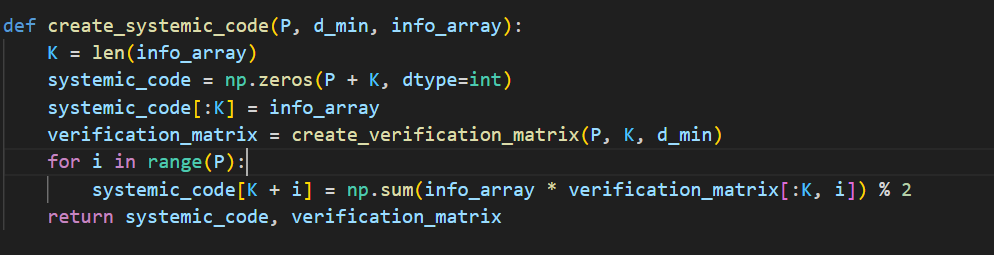
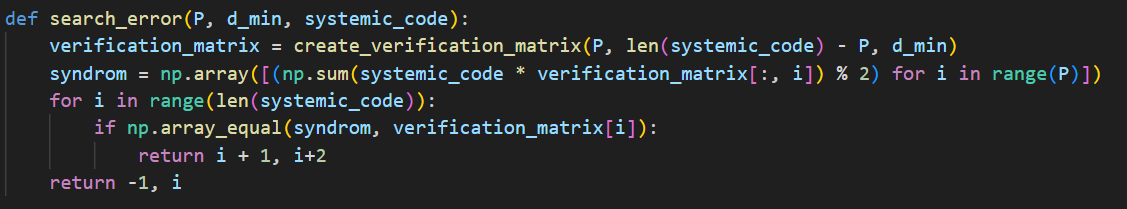
Возврат результата: возвращает готовую проверочную матрицу.

Рисунок 2 – функция create\_systematic\_code

Назначение: создаёт систематический код на основе входного информационного массива.

Как работает:

* инициализация системного кода: создаёт нулевой массив длиной P + K и копирует информационный массив в начало этого массива;
* вычисление проверочных битов: использует проверочную матрицу для вычисления проверочных битов, добавляя их к системному коду;
* возврат результата: возвращает систематический код и проверочную матрицу.

Рисунок 3 – функция search\_error

Назначение: обнаруживает позицию ошибки в систематическом коде и вычисляет синдром ошибки.

Как работает:

* создание проверочной матрицы: создаёт проверочную матрицу для систематического кода;
* вычисление синдрома ошибки: вычисляет синдром ошибки путем умножения систематического кода на проверочную матрицу и взятия остатка от деления на 2;
* поиск ошибки: сравнивает синдром с каждой строкой матрицы. Если они совпадают, возвращает позицию ошибки и синдром. Если нет, возвращает -1 и синдром;
* возврат результата: возвращает позицию ошибки и синдром.

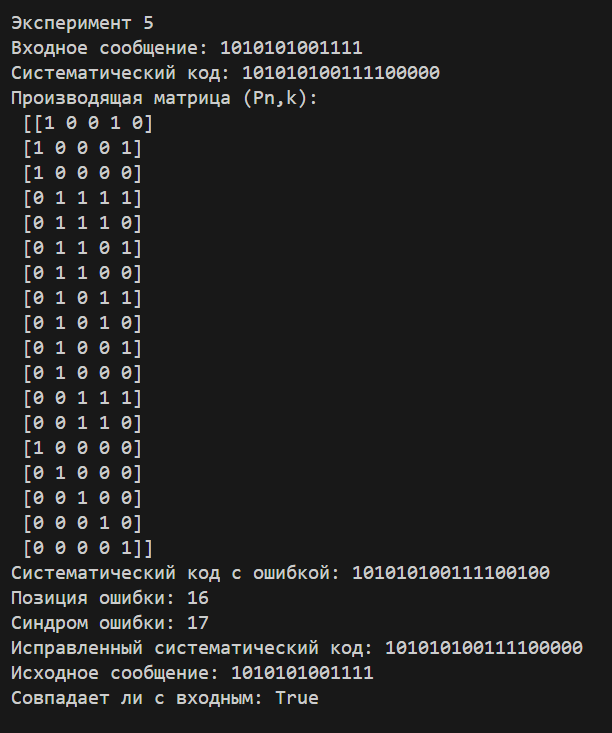


Рисунок 4 – результат работы программы

ЗАКЛЮЧЕНИЕ

В результате выполнения работы были получены навыки по формированию Систематического кода, по проверочной матрицы H и производящей матрицы H. Изучены принципы работы проверочных столбцов, алгоритм работы Систематического кода, термины позиции и синдрома ошибки

ПРИЛОЖЕНИЕ

import numpy as np

import random

def create\_verification\_matrix(P, K, d\_min):

    verification\_matrix = np.zeros((K + P, P), dtype=int)

    decrescent = P + K

    for i in range(K):

        matrix\_line = decrescent

        one\_counter = 0

        for j in range(P - 1, -1, -1):

            verification\_matrix[i, j] = matrix\_line % 2

            one\_counter += matrix\_line % 2

            matrix\_line //= 2

        if one\_counter + 1 < d\_min:

            i -= 1

        decrescent -= 1

    for i in range(K, K + P):

        for j in range(P - 1, -1, -1):

            verification\_matrix[i, j] = 1 if i - K == j else 0

    return verification\_matrix

def create\_systemic\_code(P, d\_min, info\_array):

    K = len(info\_array)

    systemic\_code = np.zeros(P + K, dtype=int)

    systemic\_code[:K] = info\_array

    verification\_matrix = create\_verification\_matrix(P, K, d\_min)

    for i in range(P):

        systemic\_code[K + i] = np.sum(info\_array \* verification\_matrix[:K, i]) % 2

    return systemic\_code, verification\_matrix

def search\_error(P, d\_min, systemic\_code):

    verification\_matrix = create\_verification\_matrix(P, len(systemic\_code) - P, d\_min)

    syndrom = np.array([(np.sum(systemic\_code \* verification\_matrix[:, i]) % 2) for i in range(P)])

    for i in range(len(systemic\_code)):

        if np.array\_equal(syndrom, verification\_matrix[i]):

            return i + 1, i+2

    return -1, i

def main():

    num\_experiments = 6

    for experiment in range(1, num\_experiments + 1):

        # Генерация случайного входного сообщения

        K = 13

        input\_str = ''.join(random.choice('01') for \_ in range(K))

        info\_array = np.array([int(bit) for bit in input\_str])

        print(f"Эксперимент {experiment}")

        print("Входное сообщение:", input\_str)

        # Создание систематического кода

        N = K

        d\_min = 2 \* 1 + 1

        while 2\*\*K > (2\*\*N / (N + 1)):

            N += 1

        systemic\_code, verification\_matrix = create\_systemic\_code(N - K, d\_min, info\_array)

        print("Систематический код:", ''.join(map(str, systemic\_code)))

        print("Производящая матрица (Pn,k):\n", verification\_matrix)

        # Внесение ошибки в случайный бит систематического кода

        error\_index = random.randint(0, N - 1)

        systemic\_code\_with\_error = systemic\_code.copy()

        systemic\_code\_with\_error[error\_index] = 1 - systemic\_code\_with\_error[error\_index]

        print("Систематический код с ошибкой:", ''.join(map(str, systemic\_code\_with\_error)))

        # Обнаружение и исправление ошибки

        error\_position, error\_syndrom = search\_error(N - K, d\_min, systemic\_code\_with\_error)

        print("Позиция ошибки:", error\_position)

        print("Синдром ошибки:", error\_syndrom)

        systemic\_code\_with\_error[error\_position - 1] = 1 - systemic\_code\_with\_error[error\_position - 1]

        print("Исправленный систематический код:", ''.join(map(str, systemic\_code\_with\_error)))

        # Вывод исходного сообщения

        output\_str = ''.join(map(str, systemic\_code\_with\_error[:K]))

        print("Исходное сообщение:", output\_str)

        print("Совпадает ли с входным:", input\_str == output\_str)

        print("\n" + "="\*50 + "\n")

if \_\_name\_\_ == "\_\_main\_\_":

    main()

Приложение 1 – листинг программного кода