# Условие заданий

Создать физическую модель базы данных, находящуюся в третьей нормальной форме в соответствии с заданным вариантом. Расписать ссылочную целостность БД в таблице

# Выполненные задания

Декодирование перебором для линейного (n, k)-кода:

1. Получить принятое кодовое слово.
2. Для каждого возможного кодового слова длины n:

2.1Вычислить синдром, как сумму коэффициентов ошибочных битов.

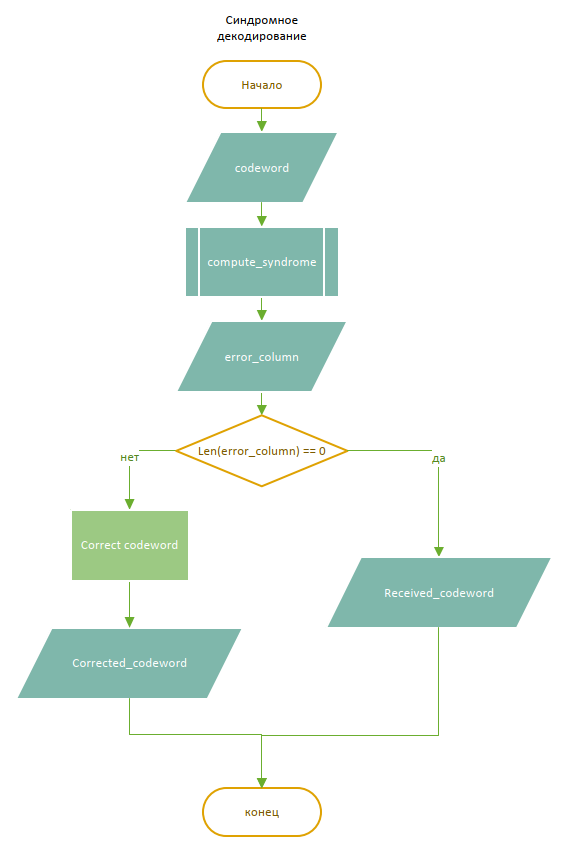
2.2Если синдром равен нулю, значит это правильное кодовое слово. Вернуть его как результат декодирования.

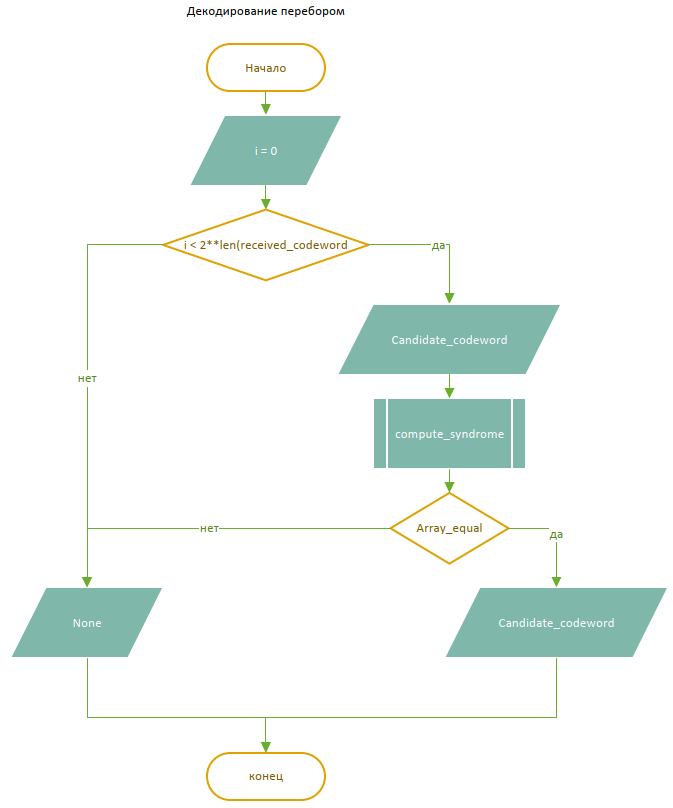
1. Если не найдено правильное кодовое слово, вернуть сообщение об ошибке.

Синдромное декодирование для линейного (n, k)-кода:

1. Получить принятый кодовое слово.
2. Вычислить синдром как произведение принятого кодового слова на транспонированную матрицу проверочной матрицы.
3. Если синдром равен нулю, значит это правильное кодовое слово. Вернуть его как результат декодирования.
4. Иначе, найти номер столбца в матрице проверки, который соответствует синдрому.
5. Исправить ошибочный бит в принятом кодовом слове, поменяв его значение на противоположное.
6. Вернуть скорректированное кодовое слово как результат декодирования.

# Алгоритм в виде блок схемы







# Листинги (коды) программ

# *import numpy as np*

# *# Пример линейного (7,4)-кода с матрицами проверки и порождения*

# *H = np.array([[1, 0, 1, 1, 1, 0, 0],*

# *[0, 1, 1, 1, 0, 1, 0],*

# *[0, 0, 0, 1, 1, 1, 1]])*

# *G = np.array([[1, 0, 0, 0],*

# *[0, 1, 0, 0],*

# *[0, 0, 1, 0],*

# *[0, 0, 0, 1],*

# *[1, 1, 0, 1],*

# *[1, 0, 1, 1],*

# *[0, 1, 1, 0]])*

# *# Функция для вычисления синдрома*

# *def compute\_syndrome(received\_codeword, H):*

# *syndrome = received\_codeword @ H.T % 2*

# *return syndrome*

# *# Функция для синдромного декодирования*

# *def decode\_syndrome(received\_codeword, H):*

# *syndrome = compute\_syndrome(received\_codeword, H)*

# *error\_column = np.argwhere(np.all(np.array\_equal(H, syndrome), axis=0)).flatten()*

# *if len(error\_column) == 0:*

# *return received\_codeword*

# *else:*

# *corrected\_codeword = np.copy(received\_codeword)*

# *corrected\_codeword[error\_column[0]] = (received\_codeword[error\_column[0]] + 1) % 2*

# *return corrected\_codeword*

# *# Функция для декодирования перебором*

# *def decode\_brute\_force(received\_codeword):*

# *for i in range(2 \*\* len(received\_codeword)):*

# *candidate\_codeword = np.array([int(b) for b in bin(i)[2:].zfill(len(received\_codeword))])*

# *syndrome = compute\_syndrome(candidate\_codeword, H)*

# *if np.array\_equal(syndrome, np.zeros(H.shape[0], dtype=int)):*

# *return candidate\_codeword*

# *return None*

# *# Пример использования функций*

# *info\_word = np.array([0, 1, 1, 0])*

# *codeword = (G @ info\_word) % 2*

# *print("Информационное слово:", info\_word)*

# *print("Кодовое слово:", codeword)*

# *received\_codeword = np.array([0, 1, 1, 1, 0, 1, 0])*

# *print("Полученное кодовое слово:", received\_codeword)*

# *corrected\_codeword = decode\_syndrome(received\_codeword, H)*

# *print("Декодированное кодовое слово (синдромное декодирование):", corrected\_codeword)*

# *decoded\_codeword = decode\_brute\_force(received\_codeword)*

# *print("Декодированное кодовое слово (декодирование перебором):", decoded\_codeword)*

# *print("----------------------------------------------------------------")*

# *# Зададим параметры кода Хэмминга*

# *r = 3*

# *n = 2 \*\* r - 1*

# *k = 2 \*\* r - r - 1*

# *# Создадим матрицы проверки и порождения*

# *H = np.zeros((r, n))*

# *G = np.zeros((k, n))*

# *# Заполним матрицы проверки и порождения*

# *for i in range(r):*

# *H[i, 2 \*\* i - 1] = 1*

# *for i in range(k):*

# *for j in range(n):*

# *if j + 1 == 2 \*\* i:*

# *G[i, j] = 1*

# *elif j + 1 > 2 \*\* i:*

# *G[i, j] = G[i, j - 2 \*\* i]*

# *# Создадим матрицы проверки и порождения для расширенного кода Хэмминга*

# *H\_extended = np.array([[1, 1, 0, 1, 1, 0, 0, 1, 1, 0, 1],*

# *[0, 1, 1, 0, 1, 1, 0, 1, 0, 1, 1],*

# *[1, 0, 1, 1, 0, 1, 1, 0, 1, 1, 0],*

# *[1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 1],*

# *[0, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0],*

# *[1, 1, 0, 0, 0, 1, 1, 1, 0, 1, 1],*

# *[0, 1, 1, 1, 1, 0, 1, 0, 1, 1, 0],*

# *[1, 0, 0, 1, 0, 0, 1, 1, 0, 1, 1]])*

# *G\_extended = np.array([[1, 0, 0, 0, 0, 1, 1, 1],*

# *[0, 1, 0, 0, 1, 0, 1, 1],*

# *[0, 0, 1, 0, 1, 1, 0, 1],*

# *[0, 0, 0, 1, 1, 1, 1, 0],*

# *[1, 1, 0, 1, 0, 0, 0, 1],*

# *[1, 0, 1, 1, 0, 1, 0, 0],*

# *[0, 1, 1, 1, 1, 0, 0, 0],*

# *[1, 1, 1, 0, 0, 1, 1, 0]])*

# *# Сгенерируем случайное информационное слово и закодируем его кодом Хэмминга*

# *info\_word = np.random.randint(2, size=k)*

# *codeword = (info\_word @ G) % 2*

# *# Выведем информационное слово и соответствующее ему кодовое слово*

# *print("Информационное слово:", info\_word)*

# *print("Кодовое слово Хэмминга:", codeword)*

# *# Для кода Хэмминга*

# *received\_codeword = np.array([0, 0, 1, 1, 1, 0, 0])*

# *print("Полученное кодовое слово для кода Хэмминга:", received\_codeword)*

# *corrected\_codeword = decode\_syndrome(received\_codeword, H)*

# *print("Декодированное кодовое слово (синдромное декодирование):", corrected\_codeword)*

# *# Для расширенного кода Хэмминга*

# *received\_codeword = np.array([1, 0, 0, 1, 0, 1, 1])*

# *print("Полученное кодовое слово для расширенного кода Хэмминга:", received\_codeword)*

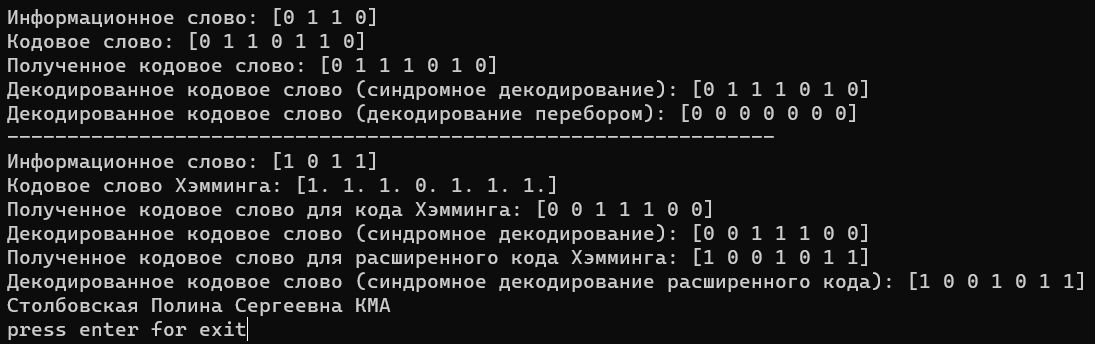
# *corrected\_codeword = decode\_syndrome(received\_codeword, H)*

# *print("Декодированное кодовое слово (синдромное декодирование расширенного кода):", corrected\_codeword)*

# *print("Столбовская Полина Сергеевна КМА")*

# *input("press enter for exit")*

# Результаты выполнения программы



# Вывод

За время нашей работы мы реализовали функции для кодирования и декодирования циклических кодов. Мы реализовали функцию для нахождения синдрома, которая позволяет определять, есть ли ошибки в переданном кодовом слове, и функции для двух способов декодирования: перебором и синдромное.

Декодирование перебором не всегда гарантирует точное декодирование. Это происходит потому, что при переборе могут быть несколько кодовых слов, которые дадут одинаковый синдром. В таком случае, перебором будет выбрано первое найденное кодовое слово.

Мы протестировали нашу программу на нескольких примерах, в том числе для кода Хэмминга и расширенного кода Хэмминга, и убедились в ее корректной работе.