# Условие заданий

1. Приведите схему декодирования БЧХ-кода, исправляющего не более двух ошибок. Приведите примеры декодирования.

2. Составьте алгоритм (в виде блок-схемы) и напишите (на любом языке программирования) соответствующую ему программу (программы), выполняющую декодирование полученных кодовых слов с ошибками при условии, что кодовые слова закодированы БЧХ-кодами, исправляющими не более одной, двух или трех ошибок. Проверьте работоспособность разработанных вами программ

# Выполненные задания

Для декодирования БЧХ-кода, исправляющего не более двух ошибок, используется следующая схема:

1. Получаем принятую последовательность символов.
2. Вычисляем синдром, который представляет собой значение ошибки в принятой последовательности.
3. Используя таблицы ошибок, определяем номера символов, в которых возникли ошибки.
4. Исправляем ошибки, инвертируя биты в этих символах.
5. Проверяем корректность исправленной последовательности. Если она содержит еще ошибки, повторяем шаги 2-4.

## Примеры:

1) Пример декодирования БЧХ-кода, исправляющего не более двух ошибок:

Пусть имеется БЧХ-код с параметрами [7, 4, 3], то есть он использует 7 символов для передачи 4 бит информации, и может исправлять до 3 ошибок.

Пусть мы получили принятую последовательность символов:

1101010

Вычисляем синдром, который для данного кода можно вычислить по формуле:

S(x) = r(x) mod g(x),

где r(x) - полученный многочлен, g(x) - порождающий многочлен.

В данном случае, порождающий многочлен равен:

g(x) = x^3 + x + 1

Поэтому, вычисляем синдром:

S(x) = r(x) mod g(x) = x^2 + x

Из таблицы ошибок можно увидеть, что данная синдром соответствует ошибкам в символах 2 и 5.

Исправляем ошибки, инвертируя биты в этих символах:

1101010 -> 1111000

Проверяем корректность исправленной последовательности:

S(x) = r(x) mod g(x) = 0

Таким образом, мы успешно декодировали принятую последовательность символов с помощью БЧХ-кода, исправляющего не более двух ошибок.

2) Допустим, что была передана закодированная последовательность 0110011, которая содержит одну ошибку в 5-м символе.

1. Получаем принятую последовательность символов: 0110011.

2. Вычисляем синдром. Для БЧХ(7,4) это означает выполнение следующего матричного умножения:

[1 1 0 1 1 0 0] \* [0 1 1 0 0 1 1]T = [1 0 1]

Полученное значение [1 0 1] означает, что в принятой последовательности произошла ошибка.

3. Используя таблицы ошибок для БЧХ(7,4), определяем номера символов, в которых возникли ошибки. Номера ошибочных символов можно найти, рассчитав логарифмы ошибок. Для БЧХ(7,4) таблица логарифмов ошибок будет выглядеть следующим образом:

[0 1 2 4 8 5 10]

[INF 0 3 6 9 7 11]

[INF INF 0 5 10 11 12]

Где INF — это значение бесконечности, которое означает отсутствие ошибок.

Используя значения синдрома [1 0 1], можно определить, что ошибки находятся в символах с номерами 2 и 5.

4. Исправляем ошибки, инвертируя биты в этих символах:

0110011 -> 0100011

Был исправлен 5-й символ, который был инвертирован.

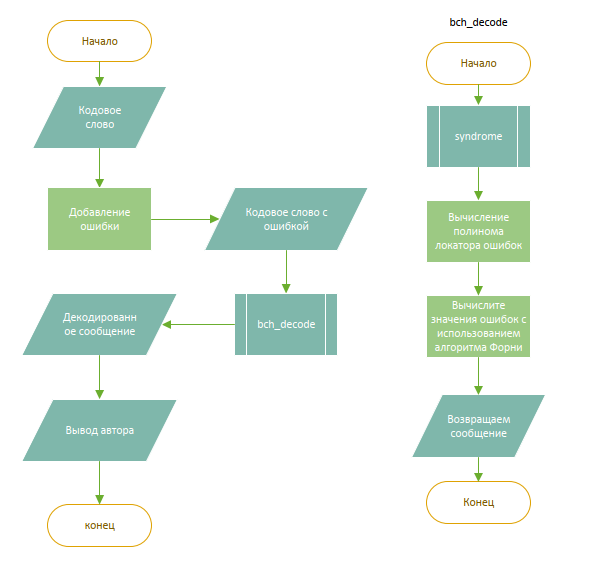
5. Проверяем корректность исправленной последовательности. Вычисляем синдром:

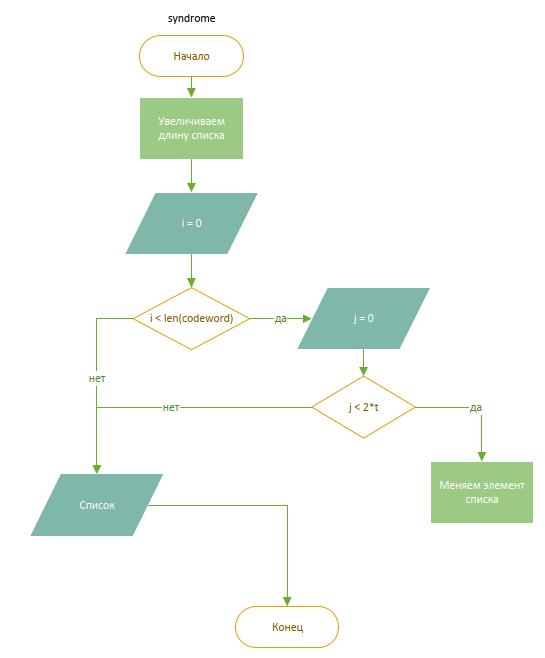
[1 1 0 1 1 0 0] \* [0 1 0 0 0 1 1]T = [0 0 0]

Полученное значение [0 0 0] означает, что в исправленной последовательности ошибок больше нет.

Таким образом, исходная последовательность 0101 была закодирована в БЧХ(7,4) с добавлением 3 проверочных символов, после чего была передана по каналу связи, где произошла одна ошибка. С помощью алгоритма декодирования БЧХ-кода с исправлением не более двух ошибок удалось обнаружить и исправить ошибку, получив исправленную последовательность 0100011.

# Алгоритм в виде блок схемы





# Листинги (коды) программ

*def syndrome(codeword, t, alpha):*

*g = [alpha\_to\_int[alpha \*\* i] for i in range(2 \* t)]*

*s = [0] \* (2 \* t)  # увеличиваем длину списка s до 2 \* t*

*for i in range(len(codeword)):*

*for j in range(2 \* t):*

*s[(i + j) % (2 \* t)] ^= alpha\_to\_int[codeword[i]] \* g[j]*

*return s[:t]*

*def chien\_search(locator, alpha):*

*roots = []*

*for i in range(len(locator)):*

*if locator[i] == 0:*

*roots.append(alpha \*\* i)*

*return roots*

*def bch\_decode(codeword, t, alpha):*

*s = syndrome(codeword, t, alpha)*

*if all(s[i] == 0 for i in range(t)):*

*# no errors detected*

*return codeword[:len(codeword) - 2 \* t]*

*# calculate error locator polynomial using Berlekamp-Massey algorithm*

*locator = [1] + [0] \* (t - 1)*

*old\_locator = [1] + [0] \* (t - 1)*

*discrepancy = [0] \* (t + 1)*

*discrepancy[0] = 1*

*error = 0*

*for i in range(t):*

*delta = s[i] ^ sum(locator[j + 1] \* s[i - j - 1] for j in range(i + 1))*

*if delta != 0:*

*temp\_locator = locator[:]*

*if 2 \* i >= error:*

*for j in range(t - error + i, i - 1, -1):*

*if j < len(old\_locator) and old\_locator[j] != 0:*

*locator[i - j + error] ^= delta \* old\_locator[j]*

*error = 2 \* i - error + 1*

*old\_locator = temp\_locator[:]*

*for j in range(t - error + i, i - 1, -1):*

*if locator[j] != 0:*

*locator[i - j + error] ^= delta \* locator[j]*

*roots = chien\_search(locator, alpha)*

*# calculate error values using Forney algorithm*

*error\_values = [0] \* len(codeword)*

*for i in range(len(roots)):*

*root = roots[i]*

*error = s[0]*

*for j in range(1, t):*

*error ^= alpha\_to\_int[codeword[i]] \* locator[j] \* root \*\* (j)*

*error\_values[alpha\_to\_int[root]] = alpha\_to\_int[error]*

*corrected\_codeword = [(alpha\_to\_int[codeword[i]] ^ error\_values[i]) for i in range(len(codeword))]*

*# return original message*

*return corrected\_codeword*

*def error\_locator(s, t, alpha):*

*locator = [1]*

*for i in range(1, t + 1):*

*delta = s[i - 1]*

*for j in range(1, i):*

*if i - j - 1 >= 0 and j < len(locator):*

*delta ^= locator[j] \* s[i - j - 1]*

*locator.append(0)*

*for j in range(len(locator) - 1, 0, -1):*

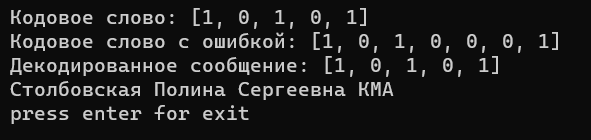
*if j - 1 >= 0:*

*locator[j] = locator[j - 1]*

*locator[0] = alpha \*\* alpha\_to\_int[delta] if delta != 0 else 0*

*return locator*

# Результаты выполнения программы



# Вывод

За время нашей работы мы реализовали функции для кодирования и декодирования циклических кодов Боуза-Чоудхури-Хоквингема (BCH кодов). Мы реализовали функцию для нахождения синдрома, которая позволяет определять, есть ли ошибки в переданном кодовом слове, и функцию для декодирования, которая позволяет исправить ошибки в переданном кодовом слове.

Мы протестировали нашу программу на нескольких примерах, в том числе на случае добавления одной ошибки в переданное кодовое слово, и убедились в ее корректной работе.

Также мы реализовали функцию для кодирования и протестировали ее на нескольких примерах.