Лабораторная работа N 4
Тема: «Избыточное кодирование данных в информационных системах. Код Хемминга»

Цель: приобретение практических навыков кодирования/декодирования двоичных данных при использовании кода Хемминга.

Задачи:

1. Закрепить теоретические знания по использованию методов помехоустойчивого кодирования для повышения надежности передачи и хранения в памяти компьютера двоичных данных.
2. Разработать приложение для кодирования/декодирования двоичной информации кодом Хемминга с минимальным кодовым расстоянием 3 или 4.
3. Результаты выполнения лабораторной работы оформить в виде отчета с листингом разработанного приложения, методики выполнения экспериментов с использованием приложения и результатов эксперимента.
4. Ответить на контрольные вопросы

1. ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

1.1. ОСНОВНЫЕ ПОНЯТИЯ И ОПРЕДЕЛЕНИЯ

Надежность системы – характеристика способности программного, аппаратного, аппаратно-программного средства выполнить при определенных условиях требуемые функции в течение конкретного периода времени.

Достоверность работы системы (устройства) – свойство, характеризующее истинность конечного (выходного) результата работы (выполнения программы), определяемое способностью средств контроля фиксировать правильность или ошибочность работы.

Ошибка устройства – неправильное значение сигнала (бита – в цифровом устройстве) на внешних выходах устройства или отдельного его узла, вызванное технической неисправностью, или воздействующими на него помехами (преднамеренными либо непреднамеренными), или иным способом.

Ошибка программы – проявляется в не соответствующем реальному (требуемому) промежуточном или конечном значении (результате) вследствие неправильно запрограммированного алгоритма или неправильно составленной программы.

Как следует из вышеприведенного определения, надежность есть внутреннее свойство объекта, заложенное в него при изготовлении и проявляющееся во время эксплуатации. Вторая особенность надежности состоит в том, что она проявляется во времени.

И третья особенность надежности выражается по-разному при различных условиях эксплуатации и различных режимах применения объекта (информационной системы в целом, отдельного ее блока, канала передачи сообщения, оперативной или внешней памяти компьютера).

Надежность является комплексным свойством, включающим в себя единичные свойства: безотказность, ремонтопригодность, сохраняемость, долговечность.

Безотказность – это свойство технического объекта непрерывно сохранять работоспособное состояние в течение некоторого времени (или наработки). Наработка, как правило, измеряется в единицах времени.

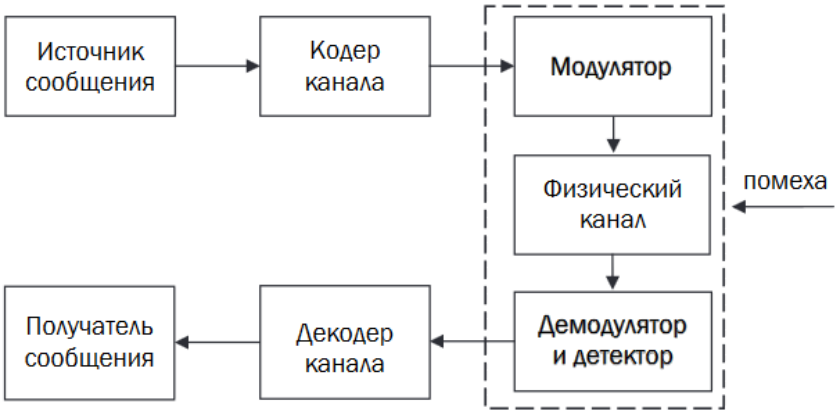
Ремонтопригодность – это свойство технического объекта, заключающееся в приспособленности к поддержанию и восстановлению работоспособного состояния путем технического обслуживания, ремонта (или с помощью дополнительных, избыточных технических средств, функционирующих параллельно с объектом).

Большинство современных цифровых систем и устройств (в том числе компьютеры и компьютерные системы, отдельные блоки и модули компьютеров – полупроводниковая, магнитная или оптическая память) содержат специальные средства, призванные автоматически восстанавливать работоспособность этих объектов при нарушении нормального функционирования.

Такие специальные средства контроля называются *избыточными*. На рисунке приведена упрощенная структурная схема системы передачи данных с избыточными средствами аппаратного контроля.

Не вдаваясь в детали функционирования блоков, названия которых на схеме выделены полужирным, рассмотрим основные принципы работы представленной схемы в соответствии с задачами данной лабораторной работы. Как видим, сначала осуществляется формирование данных в виде двоичных символов. Затем кодер канала вносит в принятую информационную последовательность некоторую *избыточность* (данный процесс называется *кодированием* или *помехоустойчивым кодированием*), которую декодер может использовать для исправления возникающих при передаче данных по каналу связи ошибок.

Таким образом, простейшая структурная схема дополнена двумя интересующими нас блоками: кодером (канала), осуществляющим преобразование исходного сообщения (*информационного слова*) (длина сообщения – символов) в избыточное сообщение (*кодовое слово*) длиной n символов (>), и декодером (канала).



Структурная схема системы передачи данных
с избыточными средствами аппаратного контроля

Изначальной причиной нарушения нормальной работы цифрового устройства являются технические *дефекты* (неисправности), возникающие внутри узлов или блоков устройства либо в каналах связи между ними.

Дефекты или неисправности могут приводить либо к кратковременному нарушению достоверности работы устройства (*сбой*), либо к полной и окончательной потере достоверности *(отказ*).

В каждом из этих случаев следствием неисправности являются ошибки в информации (*информационные ошибки*). Чаще всего причиной ошибок бывают внешние помехи, как это показано на рисунке.

Количество таких ошибок (количество ошибочных двоичных символов) принято называть *кратностью ошибки*.

Пример. Исходная (правильная) информационная последовательность = 1001. Длина этой последовательности равна 4 битам (= 4). Некоторая из перечисленных причин привела к тому, что в этой последовательности появились две ошибки (кратность ошибки равна двум): = 1111. Здесь – сообщение на выходе канала (без учета его кодирования/декодирования).

Обнаружение и/или исправление подобных ошибок как раз и призваны обеспечить кодер и декодер.

При использовании избыточных кодов исходные данные делятся на блоки из битов (называются информационными битами). В процессе кодирования каждый - битный блок данных преобразуется, как было отмечено выше, в блок из битов (кодовое слово). Число часто называется

размерностью кода. Таким образом, к каждому блоку данных в процессе кодирования присоединяются битов, которые называют *избыточными битами* (redundant bits), *битами четности* (parity bits) или *контрольными битами* (check bits); новой информации они не несут.

Для обозначения описанного кода обычно пользуются записью и говорят, что данный код использует символов для передачи (хранения) символов сообщения. Отношение числа битов данных к общему числу битов именуется степенью кодирования (code rate) – доля кода, которая приходится на полезную информацию. Еще одним важным параметром кода является *расстояние Хемминга* ( ), которое показывает, что два кодовых слова различаются по крайней мере в позициях. В общем случае код позволяет обнаруживать ошибок:

$$t_{\text{о}} = \begin{cases} \dfrac{d}{2}, & d - \text{четное}; \\ \dfrac{d-1}{2}, & d - \text{нечетное}. \end{cases}$$

Количество исправляемых кодом ошибок определяется следующим образом:

$$t_{\text{и}} = \begin{cases} \dfrac{d-1}{2}, & d - \text{нечетное}; \\ \dfrac{d-2}{2}, & d - \text{четное}. \end{cases}$$

Во всех этих простых математических выражениях d соответствует минимальному кодовому расстоянию Хемминга анализируемого кода.

1.2. ТЕОРЕТИЧЕСКИЕ ОСНОВЫ ЛИНЕЙНЫХ БЛОЧНЫХ КОДОВ

К. Шеннон сформулировал теорему для случая передачи дискретной информации по каналу связи с помехами, утверждающую, что вероятность ошибочного декодирования принимаемых сигналов может быть обеспечена сколь угодно малой путем выбора соответствующего способа кодирования сигналов. В теореме Шеннона не говорится о том, как нужно строить необходимые помехоустойчивые коды. Однако в ней указывается на принципиальную возможность кодирования, при котором может быть обеспечена сколь угодно высокая надежность передачи.

Код Хемминга относится к классу линейных блочных кодов.

Линейные блочные коды – это класс кодов с контролем четности, которые можно описать парой чисел . Для формирования проверочных символов (кодирования), т. е. вычисления проверочного слова , используется порождающая матрица : совокупность базисных векторов будем далее записывать в виде матрицы размерностью с единичной под матрицей в первых строках и столбцах:

$$G = [P \mid I].$$

Более точно матрица называется *порождающей матрицей линейного корректирующего кода в приведено - ступенчатой форме*.

Кодовые слова являются линейными комбинациями строк матрицы (кроме слова, состоящего из нулевых символов). Кодирование заключается в умножении вектора сообщения длиной на порождающую матрицу по правилам матричного умножения (все операции выполняются по модулю 2). Очевидно, что при этом первые символов кодового слова равны соответствующим символам сообщения, а последние символов образуются как линейные комбинации первых.

Для всякой порождающей матрицы существует матрица размерности , задающая базис нулевого пространства кода и удовлетворяющая равенству

$$G \cdot H^T = 0.$$

Справедливо также

$$X_n \cdot H^T = H \cdot (X_n)^T = 0.$$

В последнем выражении символ « » означает *транспонирование*, а

Матрица , называемая *проверочной*, равна

$$H = \left[-P^T \mid I \right] = P' \mid I.$$

В коде Хемминга с минимальным кодовым расстоянием проверочная матрица имеет классический вид и состоит из двух подматриц: размером и размером соответственно.

В последнем выражении – единичная матрица порядка . Количество избыточных (проверочных) символов кодового слова определяется из следующей простой логической цепи рассуждений.

Общее число всех возможных комбинаций должно удовлетворять неравенству

$$2^r \geq n+1$$

в силу того, что

$$2^r \geq n+1 \Rightarrow 2^r \geq k+r+1 \Rightarrow r \geq \log_2(k+r+1) \Rightarrow$$
$$r \geq \log_2(k+1).$$

Присутствие цифры «1» в приведенных выражениях соотносите с нулевым вектор - столбцом, который в матрице не используется.

Например, для (7,4) - кода Хемминга проверочная матрица в упорядоченном виде выглядит так:

$$H_{(7,4)} = \begin{vmatrix} 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 \end{vmatrix}.$$

Результат умножения сообщения на выходе канала передачи ( ) или (что равнозначно) сообщения, считываемого из памяти, на проверочную матрицу ( ) называется **синдромом (вектором ошибки)** S:

$$S = H \cdot (Y_n)^T = Y_n \cdot H^T,$$

где – принятый вектор (сообщение на выходе канала), полученный после передачи либо считывания из памяти.

Вектор обычно представляют в следующем виде:

$$Y_n = X_n + E_n,$$

где , – вектор ошибки.

Синдром – это результат проверки четности, выполняемой над сообщением для определения его принадлежности заданному набору кодовых слов. При положительном результате проверки синдром равен 0, т.е. . Если содержит ошибки, которые можно исправить, то синдром имеет определенное ненулевое значение, что позволяет обнаружить и исправить конкретную ошибочную комбинацию.

Важно запомнить, что ненулевой синдром всегда равен сумме по модулю 2 тех вектор столбцов матрицы , номера которых соответствуют номерам ошибочных битов в слове .

Корректирующая способность кода Хемминга с может быть увеличена введением дополнительной проверки на четность.

В этом случае проверочная матрица будет иметь вид:

$$H' = \left|\begin{array}{ccc:c} & & & 0 \\ & H & & 0 \\ & & & \vdots \\ & & & 0 \\ \hdashline 1 & 1 & \cdots\ 1 & 1 \end{array}\right|$$

Так, минимальное кодовое расстояние такого кода будет равно 4: . Такой код может исправлять все единичные ошибки с одновременным обнаружением всех двойных в анализируемом кодовом слове.

При этом нужно помнить, что вид матрицы не соответствует ее каноническому представлению, поскольку во всех столбцах единичной матрицы, кроме последнего, будет по 2 единицы.

Для придания матрице канонического вида необходимо сложить посимвольно все строки между собой и результат сложения записать в последнюю строку (под горизонтальной линией).

2. ПРАКТИЧЕСКИЙ ПРИМЕР

Построение (7, 4)-кода Хемминга

Вернемся к Хеммингу. Слова (7, 4)-кода образованы из 7 разрядов С j = $(i_1, i_2, i_3, i_4, p_1, p_2, p_3)$, j = 0(1)15, 4-информационные и 3-проверочные символа, т.е. по существу избыточные, так как они не несут информации сообщения. Эти три проверочных разряда удалось представить линейными функциями 4-х информационных символов в каждом слове, что и обеспечило обнаружение факта ошибки и ее места в словах, чтобы внести исправление. А (7, 4)-код получил новое прилагательное и стал *линейным блоковым двоичным*.

Линейные функциональные зависимости (правила (*)) вычислений значений символов p_i имеют следующий вид:

$$p_1 + i_2 + i_3 + i_4 = 0, p_1 = i_2 + i_3 + i_4,$$
$$p_2 + i_1 + i_3 + i_4 = 0, p_2 = i_1 + i_3 + i_4, (*)$$
$$p_3 + i_1 + i_2 + i_4 = 0, p_3 = i_1 + i_2 + i_4.$$

Исправление ошибки стало очень простой операцией — в ошибочном разряде определялся символ (ноль или единица) и заменялся другим противоположным 0 на 1 или 1 на 0.
Сколько же различных слов образуют код? Ответ на этот вопрос для (7, 4)-кода получается очень просто. Раз имеется лишь 4 информационных разряда, а их разнообразие при заполнении символами имеет $_2{}^4$ = 16 вариантов, то других возможностей просто нет, т. е. код состоящий всего из 16 слов, обеспечивает представление этими 16-ю словами всю письменность всего языка.

Информационные части этих 16 слов получают нумерованный вид № (i_1, i_2, i_3, i_4):

0=0000; 4= 0100; 8=1000; 12=1100;
1=0001; 5= 0101; 9=1001; 13=1101;
2=0010; 6= 0110; 10=1010; 14=1110;
3=0011; 7= 0111; 11=1011; 15=1111.

Каждому из этих 4-разрядных слов необходимо вычислить и добавить справа по 3 проверочных разряда, которые вычисляются по правилам (*). Например, для информационного слова №6 равного 0110 имеем $i_1 = 0, i_2 = 1, i_3 = 1, i_4 = 0$ и вычисления проверочных символов дают для этого слова такой результат:

$$(_1 = 0, _2 = 1, _3 = 1)$$
$$_1 = i_2 + i_3 + i_4 = 1 + 1 + 0(mod2) = 2(mod2) = 0,$$
$$_2 = i_1 + i_3 + i_4 = 0 + 1 + 0(mod2) = 1(mod2) = 1,$$
$$_3 = i_1 + i_2 + i_4 = 0 + 1 + 0(mod2) = 1(mod2) = 1.$$

Шестое кодовое слово при этом приобретает вид: $_6 = (i_1, i_2, i_3, i_4, p_1, p_2, p_3) = (0110011)$. Таким же образом необходимо вычислить проверочные символы для всех 16-и кодовых слов. Подготовим для слов кода 16-строчную таблицу К и последовательно будем заполнять ее клетки (читателю рекомендую проделать это с карандашом в руках).

поиска

Таблица К – кодовые слова Cj, j = 0(1)15, (7, 4) – кода Хемминга

Номер по пор.	Дес №	Информационные				Проверочные			Вес
		i_1	i_2	i_3	i_4	P_1	P_2	P_3	W_R
0	0	0	0	0	0	0	0	0	0
1	15	0	0	0	1	1	1	1	4
2	22	0	0	1	0	1	1	0	3
3	25	0	0	1	1	0	0	1	3
4	37	0	1	0	0	1	0	1	3
5	42	0	1	0	1	0	1	0	3
6	51	0	1	1	0	0	1	1	4
7	60	0	1	1	1	1	0	0	4
8	67	1	0	0	0	0	1	1	3
9	76	1	0	0	1	1	0	0	3
10	85	1	0	1	0	1	0	1	4
11	90	1	0	1	1	0	1	0	4
12	102	1	1	0	0	1	1	0	4
13	105	1	1	0	1	0	0	1	4
14	112	1	1	1	0	0	0	0	3
15	127	1	1	1	1	1	1	1	7

Описание таблицы: 16 строк — кодовые слова; 10 колонок: порядковый номер, десятичное представление кодового слова, 4 информационных символа, 3 проверочных символа, W-вес кодового слова равен числу ненулевых разрядов (≠ 0). Заливкой выделены 4 кодовых слова-строки — это базис векторного подпространства. Собственно, на этом все — код построен.

Таким образом, в таблице получены все слова (7, 4) — кода Хемминга. Как видите это было не очень сложно. Далее речь пойдет о том, какие идеи привели Хемминга к такому построению кода. Мы все знакомы с кодом Морзе, с флотским семафорным алфавитом и др. системами построенными на разных эвристических принципах, но здесь в (7, 4)-коде используются впервые строгие математические принципы и методы. Рассказ будет как раз о них.

Математические основы кода. Высшая алгебра

Подошло время рассказать какая Р.Хеммингу пришла идея открытия такого кода. Он не питал особых иллюзий о своем таланте и скромно формулировал перед собой задачу: создать код, который бы обнаруживал и исправлял в каждом слове одну ошибку (на деле обнаруживать удалось даже две ошибки, но исправлялась лишь одна из них). При качественных каналах даже одна ошибка — редкое событие. Поэтому замысел Хемминга все-таки в масштабах системы связи был грандиозным. В теории кодирования после его публикации произошла революция.

Это был 1950 год. Я привожу здесь свое простое (надеюсь доступное для понимания) описание, которого не встречал у других авторов, но как оказалось, все не так просто. Потребовались знания из многочисленных областей математики и время, чтобы все глубоко осознать и самому понять, почему это так сделано. Только после этого я смог оценить ту красивую и достаточно простую идею, которая реализована в этом корректирующем коде. Время я в основном, потратил на разбирательство с техникой вычислений и теоретическим обоснованием всех действий, о которых здесь пишу.

Создатели кодов, долго не могли додуматься до кода, обнаруживающего и исправляющего две ошибки. Идеи, использованные Хеммингом, там не срабатывали. Пришлось искать, и нашлись новые идеи. Очень интересно! Захватывает. Для поиска новых идей потребовалось около 10 лет и только после этого произошел прорыв. Коды, обнаруживающие произвольное число ошибок, были получены сравнительно быстро.

Векторные пространства, поля и группы. Полученный (7, 4)-код (Таблица К) представляет множество кодовых слов, являющихся элементами векторного подпространства (порядка 16, с размерностью 4), т.е. частью векторного пространства размерности 7 с порядком $2^7 = 128.$ Из 128 слов в код включены лишь 16, но они попали в состав кода не просто так.

Векторные пространства, поля и группы. Полученный (7, 4)-код (Таблица К) представляет множество кодовых слов, являющихся элементами векторного подпространства (порядка 16, с размерностью 4), т.е. частью векторного пространства размерности 7 с порядком $2^7 = 128.$ Из 128 слов в код включены лишь 16, но они попали в состав кода не просто так.

Во-первых, они являются подпространством со всеми вытекающими отсюда свойствами и особенностями, во-вторых, кодовые слова являются подгруппой большой группы порядка 128, даже более того, аддитивной подгруппой конечного расширенного поля Галуа GF(2^7) степени расширения n = 7 и характеристики 2. Эта большая подгруппа раскладывается в смежные классы по меньшей подгруппе, что хорошо иллюстрируется следующей таблицей Г. Таблица разделена на две части: верхняя и нижняя, но читать следует как одну длинную. Каждый смежный класс (строка таблицы) — элемент факторгруппы по эквивалентности составляющих.

Таблица Г – Разложение аддитивной группы поля Галуа GF (2^7) в смежные классы (строки таблицы Г) по подгруппе 16 порядка.

Сферы радиуса $r = 1$, центры сфер – слова кода (верхняя строчка)								
Синдром	**Лидеры 0 класса**	**15**	**22**	**25**	**37**	**42**	**51**	**60**
000	**0000000**	**0001111**	**0010110**	**0011001**	**0100101**	**0101010**	**0110011**	**0111100**
001	**0000001**	**0001110**	**0010111**	**0011000**	**0100100**	**0101011**	**0110010**	**0111101**
010	**0000010**	**0001101**	**0010100**	**0011011**	**0100111**	**0101000**	**0110001**	**0111110**
011	**0000100**	**0001011**	**0010010**	**0011101**	**0100001**	**0101110**	**0110111**	**0111000**
100	**0001000**	**0000111**	**0011110**	**0010001**	**0101101**	**0100010**	**0111011**	**0110100**
101	**0010000**	**0011111**	**0000110**	**0001001**	**0110101**	**0111010**	**0100011**	**0101100**
110	**0100000**	**0101111**	**0110110**	**0111001**	**0000101**	**0001010**	**0010011**	**0011100**
111	**1000000**	**1001111**	**1010110**	**1011001**	**1100101**	**1101010**	**1110011**	**1111100**
	67	**76**	**85**	**90**	**102**	**105**	**112**	**127**
000	**1000011**	**1001100**	**1010101**	**1011010**	**1100110**	**1101001**	**1110000**	**1111111**
001	**1000010**	**1001101**	**1010100**	**1011011**	**1100111**	**1101000**	**1110001**	**1111110**
010	**1000001**	**1001110**	**1010111**	**1011000**	**1100100**	**1101011**	**1110010**	**1111101**
011	**1000111**	**1001000**	**1010001**	**1011110**	**1100010**	**1101101**	**1110100**	**1111011**
100	**1001011**	**1000100**	**1011101**	**1010010**	**1101110**	**1100001**	**1111000**	**1110111**
101	**1010011**	**1011100**	**1000101**	**1001010**	**1110110**	**1111001**	**1100000**	**1101111**
110	**1100011**	**1101100**	**1110101**	**1111010**	**1000110**	**1001001**	**1010000**	**1011111**
111	**0000011**	**0001100**	**0010101**	**0011010**	**0100110**	**0101001**	**0110000**	**0111111**

Столбцы таблицы – это сферы радиуса 1. Левый столбец (повторяется) – синдром слова (7, 4)-кода Хемминга, следующий столбец — лидеры смежного класса. Раскроем двоичное представление одного из элементов (25-го выделен заливкой) факторгруппы и его десятичное представление:

$$0\Delta x^6 + 0\Delta x^5 + 1\Delta x^4 + 1\Delta x^3 + 0x^2 + 0x + 1 = 1\Delta 2^4 + 1\Delta 2^3 + 1\Delta 2^0 = 16 + 8 + 1 = 25$$

Техника получение строк таблицы Г. Элемент из столбца лидеров класса суммируется с каждым элементом из заголовка столбца таблицы Г (суммирование выполняется для строки лидера в двоичном виде по mod2). Поскольку все лидеры классов имеют вес W=1, то все суммы отличаются от слова в заголовке столбца только в одной позиции (одной и той же для всей строки, но разных для столбца). Таблица Г имеет замечательную геометрическую интерпретацию. Все 16 кодовых слов представляются центрами сфер в 7-мерном векторном пространстве. Все слова в столбце от верхнего слова отличаются в одной позиции, т. е. лежат на поверхности сферы с радиусом r =1.

В этой интерпретации скрывается идея обнаружения одной ошибки в любом кодовом слове. Работа идет со сферами. Первое условие обнаружения ошибки — сферы радиуса 1 не должны касаться или пересекаться. Это означает, что центры сфер удалены друг от друга на расстояние 3 или более. При этом сферы не только не пересекаются, но и не касаются одна другой. Это требование для однозначности решения: какой сфере отнести полученное на приемной стороне декодером ошибочное (не кодовое одно из 128 -16 = 112) слово.

Второе — все множество 7-разрядных двоичных слов из 128 слов равномерно распределено по 16 сферам. Декодер может получить слово лишь из этого множества 128-ми известных слов с ошибкой или без нее. Третье — приемная сторона может получить слово без ошибки или с искажением, но всегда принадлежащее одной из 16-и сфер, которая легко определяется декодером. В последней ситуации принимается решение о том, что послано было кодовое слово — центр определенной декодером сферы, который нашел позицию (пересечение строки и столбца) слова в таблице Г, т. е номера столбца и строки.

Здесь возникает требование к словам кода и к коду в целом: расстояние между любыми двумя кодовыми словами должно быть не менее трех, т. е. разность для пары кодовых слов, например, Ci = 85=$(i_1, i_2, i_3, i_4, p_1, p_2, p_3)$=1010101; Cj = 25=$(i_1, i_2, i_3, i_4, p_1, p_2, p_3)$= 0011001 должна быть не менее 3; 85 — 25 = 1010101 — 0011001 =1001100 = 76, вес слова-разности W(76) = 3. (табл. Д заменяет вычисления разностей и сумм). Здесь под расстоянием между двоичными словами-векторами понимается количество не совпадающих позиций в двух словах. Это расстояние Хемминга, которое стало повсеместно использоваться в теории, и на практике, так как удовлетворяет всем аксиомам расстояния.

Замечание. (7, 4)-код не только *линейный блоковый двоичный*, но он еще и *групповой*, т. е. слова кода образуют алгебраическую группу по сложению. Это означает, что любые два кодовых слова при суммировании снова дают одно из кодовых слов. Только это не обычная операция суммирования, выполняется сложение по модулю два.

Таблица Д — Сумма элементов группы (кодовых слов), используемой для построения кода Хемминга

⊕	0	15	22	25	37	42	51	60	67	76	85	90	102	105	112	127
0	0	15	22	25	37	42	51	60	67	76	85	90	102	105	112	127
15	15	0	25	22	42	37	60	51	76	67	90	85	105	102	127	112
22	22	25	0	15	51	60	37	42	85	90	67	76	112	127	102	105
25	25	22	15	0	60	51	42	37	90	85	76	67	127	112	105	102
37	37	42	51	60	0	15	22	25	102	105	112	127	67	76	85	90
42	42	37	60	51	15	0	25	22	105	102	127	112	76	67	90	85
51	51	60	37	42	22	25	0	15	112	127	102	105	85	90	67	76
60	60	51	42	37	25	22	15	0	127	112	105	102	90	85	76	67

67	67	76	85	90	102	105	112	127	0	15	22	25	37	42	51	60
76	76	67	90	85	105	102	127	112	15	0	25	22	42	37	60	51
85	85	90	67	76	112	127	102	105	22	25	0	15	51	60	37	42
90	90	85	76	67	127	112	105	102	25	22	15	0	60	51	42	37
102	102	105	112	127	67	76	85	90	37	42	51	60	0	15	2	25
105	105	102	127	112	76	67	90	85	42	37	60	51	15	0	25	22
112	112	127	102	105	85	90	67	76	51	60	37	42	2	25	0	15
127	127	112	105	102	90	85	76	67	60	51	42	37	25	22	15	0

Сама операция суммирования слов ассоциативна, и для каждого элемента в множестве кодовых слов имеется противоположный ему, т. е. суммирование исходного слова с противоположным дает нулевое значение. Это нулевое кодовое слово является нейтральным элементом в группе. В таблице Д- это главная диагональ из нулей. Остальные клетки (пересечения строка/столбец) — это номера-десятичные представления кодовых слов, полученные суммированием элементов из строки и столбца.При перестановке слов местами (при суммировании) результат остается прежним, более того, вычитание и сложение слов имеют одинаковый результат. Дальше рассматривается система кодирования/декодирования, реализующая синдромный принцип.

Применение кода. Кодер

Кодер размещается на передающей стороне канала и им пользуется отправитель сообщения. Отправитель сообщения (автор) формирует сообщение в алфавите естественного языка и представляет его в цифровом виде. (Имя символа в ASCII-соде и в двоичном виде).
Тексты удобно формировать в файлах для ПК с использованием стандартной клавиатуры (ASCII — кодов). Каждому символу (букве алфавита) соответствует в этой кодировке октет бит (восемь разрядов). Для (7, 4)- кода Хемминга, в словах которого только 4 информационных символа, при кодировании символа клавиатуры на букву требуется два кодовых слова, т.е. октет буквы разбивается на два информационных слова естественного языка (ЕЯ) вида
$i < 4 >=< i_1, i_2, i_3, i_4 >$.

Пример 1. Необходимо передать слово «цифра» в ЕЯ. Входим в таблицу ASCII-кодов, буквам соответствуют: ц –11110110, и –11101000, ф – 11110100, р – 11110000, а – 11100000 октеты. Или иначе в ASCII — кодах слово «цифра» = 1111 0110 1110 1000 1111 0100 1111 0000 1110 0000

с разбивкой на тетрады (по 4 разряда). Таким образом, кодирование слова «цифра» ЕЯ требует 10 кодовых слов (7, 4)-кода Хемминга. Тетрады представляют информационные разряды слов сообщения. Эти информационные слова (тетрады) преобразуются в слова кода (по 7 разрядов) перед отправкой в канал сети связи. Выполняется это путем векторно-матричного умножения: информационного слова на порождающую матрицу. Плата за удобства получается весьма дорого и длинно, но все работает автоматически и главное — сообщение защищается от ошибок. Порождающая матрица (7, 4)-кода Хемминга или генератор слов кода получается выписыванием базисных векторов кода и объединением их в матрицу. Это следует из теоремы линейной алгебры: любой вектор пространства (подпространства) является линейной комбинацией базисных векторов, т.е. линейно независимых в этом пространстве. Это как раз и требуется — порождать любые векторы (7-разрядные кодовые слова) из информационных 4-разрядных.

Порождающая матрица (7, 4, 3)-кода Хемминга или генератор слов кода имеет вид:

$$G_{[k,n]} = \begin{bmatrix} 1\,0\,0\,0\,0\,1\,1 \\ 0\,1\,0\,0\,1\,0\,1 \\ 0\,0\,1\,0\,1\,1\,0 \\ 0\,0\,0\,1\,1\,1\,1 \end{bmatrix} \begin{matrix} 6\,7 \rightarrow \text{№ } 8 \\ 3\,7 \rightarrow \text{№ } 4 \\ 2\,2 \rightarrow \text{№ } 2 \\ 1\,5 \rightarrow \text{№ } 1 \end{matrix}$$

Справа указаны десятичные представления кодовых слов Базиса подпространства и их порядковые номера в таблице К
№ i строки матрицы — это слова кода, являющиеся базисом векторного подпространства.

Пример кодирования слов информационных сообщений (порождающая матрица кода выстраивается из базисных векторов и соответствует части таблицы К). В таблице ASCII-кода берем букву ц = <1111 0110>.

Информационные слова сообщения имеют вид:

$$i_{k1} =< 1111 >, i_{k2} =< 0110 >.$$

Это половины символа (ц). Для (7, 4)-кода, определенного ранее, требуется найти кодовые слова, соответствующее информационному слову-сообщению (ц) из 8-и символов в виде:

$$i_{k1} =< 1111 >, i_{k2} =< 0110 >.$$

Чтобы превратить эту букву–сообщение (ц) в кодовые слова u, каждую половинку буквы-сообщения i умножают на порождающую матрицу G[k, n] кода (матрица для таблицы К):

$$u_{\langle n\rangle} = i_{\langle k\rangle}(ц / 2) \times G_{[k,n]} = \langle 1111\rangle \times \begin{bmatrix} 1000011 \\ 0100101 \\ 0010110 \\ 0001111 \end{bmatrix} = \langle 1111111\rangle \rightarrow 127_{10} \rightarrow №15;$$

$$u_{\langle n\rangle} = i_{\langle k\rangle}(ц / 2) \times G_{[k,n]} = \langle 0110\rangle \times \begin{bmatrix} 1000011 \\ 0100101 \\ 0010110 \\ 0001111 \end{bmatrix} = \langle 0110011\rangle \rightarrow 51_{10} \rightarrow №6.$$

Получили два кодовых слова с порядковыми номерами 15 и 6.

Покажем детальное формирование нижнего результата №6 – кодового слова (умножение строки информационного слова на столбцы порождающей матрицы); суммирование по (mod2)

<0110> · <1000> = 0·1 +1·0 + 1·0 + 0·0 = 0(mod2);
<0110> · <0100> = 0·0 +1·1 + 1·0 + 0·0 = 1(mod2);
<0110> · <0010> = 0·0 +1·0 + 1·1 + 0·0 = 1(mod2);
<0110> · <0001> = 0·0 +1·0 + 1·0 + 0·1 = 0(mod2);
<0110> · <0111> = 0·0 +1·1 + 1·1 + 0·1 = 0(mod2);
<0110> · <1011> = 0·1 +1·0 + 1·1 + 0·1 = 1(mod2);
<0110> · <1101> = 0·1 +1·1 + 1·0 + 0·1 = 1(mod2).

Получили в результате перемножения пятнадцатое и шестое слова из таблицы К. Первые четыре разряда в этих кодовых словах (результатах умножения) представляют информационные слова. Они имеют вид: $i_{k1} =< 1111 >, i_{k2} =< 0110 >$, (в таблице ASCII это только половины буквы ц). Для кодирующей матрицы выбраны в качестве базисных векторов в таблице К совокупности слов с номерами: 1, 2, 4, 8. В таблице они выделены заливкой. Тогда для этой таблицы К кодирующая матрица получит вид G[k,n].

В результате перемножения получили 15 и 6 слова таблицы К кода.

Применение кода. Декодер

Декодер размещается на приемной стороне канала там, где находится получатель сообщения. Назначение декодера состоит в предоставлении получателю переданного сообщения в том виде, в котором оно существовало у отправителя в момент отправления, т.е. получатель может воспользоваться текстом и использовать сведения из него для своей дальнейшей работы.

Основной задачей декодера является проверка того, является ли полученное слово (7 разрядов) тем, которое было отправлено на передающей стороне, не содержит ли слово ошибок. Для решения этой задачи для каждого полученного слова декодером путем умножения его на проверочную матрицу H[n-k, n] вычисляется короткий вектор-синдром S (3 разряда).

Для слов, которые являются кодовыми, т. е. не содержат ошибок, синдром всегда принимает нулевое значение S =<000>. Для слова с ошибкой синдром не нулевой S ≠ 0. Значение синдрома позволяет обнаружить и локализовать положение ошибки с точностью до разряда в принятом на приемной стороне слове, и декодер может изменить значение этого разряда. В проверочной матрице кода декодер находит столбец, совпадающий со значением синдрома, и порядковый номер этого столбца принимает равным искаженному ошибкой разряду. После этого для двоичных кодов декодером выполняется изменение этого разряда – просто замена на противоположное значение, т. е. единицу заменяют нулем, а нуль – единицей.

Рассматриваемый код является *систематическим*, т. е. символы информационного слова размещаются подряд в старших разрядах кодового слова. Восстановление информационных слов выполняется простым отбрасыванием младших (проверочных) разрядов, число которых известно. Далее используется таблица ASCII-кодов в обратном порядке: входом являются информационные двоичные последовательности, а выходом – буквы алфавита естественного языка. Итак, (7, 4)-код *систематический, групповой, линейный, блочный, двоичный.*

Основу декодера образует проверочная матрица H[n-k, n], которая содержит число строк, равное числу проверочных символов, а столбцами все возможные, кроме нулевого, столбцы из трех символов $2^3 - 1 = 7$. Проверочная матрица строится из слов таблицы К, они выбираются так, чтобы быть ортогональными к кодирующей матрице, т.е. их произведение — нулевая матрица. Проверочная матрица получает следующий вид в операциях умножения она транспонируется. Для конкретного примера проверочная матрица H[n-k, n] приведена ниже:

$$H_{[3\times 7]} = \begin{bmatrix} 0111 & 100 \\ 1011 & 010 \\ \underbrace{1101}_{\text{информ. символы}} & \underbrace{001}_{\text{провероч. символы}} \end{bmatrix} \begin{matrix} 60 \to №7 \\ 90 \to №11 \\ 105 \to №13 \end{matrix},$$

Соотношения для проверочных символов P_1, P_2, P_3 можно переписать в векторно – матричной форме $G_{[k,n]} \times H^t_{[n-k,k]} = G_{\{4\times 7\}}\, H^t_{[3\times 7]} = 0_{<7>}$, где матрицы выписаны для таблицы K.

Матрица $H_{[n-k,\ n]}$ называется *проверочной матрицей* ее размерность $(n-k)\times n$. Число ее строк 3 (это строки таблицы с номерами $7, 11, 13$) соответствуют количеству проверочных символов, а число столбцов равно $2^3 - 1 = 7$. В матрице отсутствует столбец из трех нулей.

Далее, так как вектор-строки порождающей матрицы $G_{[k,\ n]}$ кода C являются базисными кодовыми словами, то всегда выполняется равенство $G_{[k,\ n]} \times H^t_{[n-k,\ n]} = 0_{[k,\ n-k]}$.

Покажем на примере для таблицы K выполнение этого соотношения (вычисление синдромов)

$$G_{[k,n]} \times H^{t}_{[n-k,k]} = \left\langle \begin{matrix} 1000011 \\ 0100101 \\ 0010110 \\ 0001111 \end{matrix} \right\rangle \times \begin{bmatrix} 011 \\ 101 \\ 110 \\ 111 \\ 100 \\ 010 \\ 001 \end{bmatrix} = \left\langle \begin{matrix} 000 \\ 000 \\ 000 \\ 000 \end{matrix} \right\rangle$$

<1000011>	<1000011>	<1000011>	<0100101>	<0100101>	<0100101>
<0111100>	<1011010>	<1101001>	<0111100>	<1011010>	<1101001>
0000000=0	1000010=0	1000001=0	0100100=0	0000000=0	0100001=0
<0010110>	<0010110>	<0010110>	<0001111>	<0001111>	<0001111>
<0111100>	<1011010>	<1101001>	<0111100>	<1011010>	<1101001>
0010100=0	0010010=0	0000000=0	0001100=0	0001010=0	0001001=0

Видим, что произведение порождающей матрицы на проверочную в результате дает нулевую матрицу.

Пример 2. Декодирование слова кода Хемминга без ошибки (e<7> =<0000000>).
Пусть на приемном конце канала приняты слова №7→60 и №13→105 из таблицы К,
u<7> + e<7> = <0 1 1 1 1 0 0 > + <0 0 0 0 0 0 0>,
где ошибка отсутствует, т. е. имеет вид e<7> = <0 0 0 0 0 0 0>.

$$\left(e_{\langle 7 \rangle} + u_{\langle 7 \rangle}\right) \times H^{t}_{[n-k,k]} = \langle 0111100 \rangle \times \begin{bmatrix} 011 \\ 101 \\ 110 \\ 111 \\ 100 \\ 010 \\ 001 \end{bmatrix} = \langle 000 \rangle = S_{\langle n-k \rangle}$$

Выполняем умножение вектора [1×7] на матрицу [7×3] с результатом-матрицей [1×3], которая называется синдромом кодового слова. Такое декодирование называется синдромным.

<0 1 1 1 1 0 0>	<0 1 1 1 1 0 0>	<0 1 1 1 1 0 0>
<0 1 1 1 1 0 1>	<1 0 1 1 0 1 0>	<1 1 0 1 0 0 1>
0 1 1 1 1 0 0=0	0 0 1 1 0 0 0=0	0 1 0 1 0 0 0=0

$$\left(e_{\langle 7 \rangle} + u_{\langle 7 \rangle}\right) \times H^{t}_{[n-k,k]} = \langle 1101001 \rangle \times \begin{bmatrix} 011 \\ 101 \\ 110 \\ 111 \\ 100 \\ 010 \\ 001 \end{bmatrix} = \langle 000 \rangle = S_{\langle n-k \rangle}$$

В результате вычисленный синдром имеет нулевое значение, что подтверждает отсутствие ошибки в словах кода.

Пример 3. Обнаружение одной ошибки в слове, полученном на приемном конце канала (таблица К).

А) Пусть требуется передать 7 – е кодовое слово, т.е.

u<7> = <0 1 1 1 1 0 0> и в одном третьем слева разряде слова, допущена ошибка. Тогда она суммируется по mod2 с 7-м передаваемым по каналу связи кодовым словом
u<7> + e<7> = <0 1 1 1 1 0 0 > + <0 0 1 0 0 0 0> = <0 1 0 1 1 0 0>,
где ошибка имеет вид e<7> = <0 0 1 0 0 0 0>.

Установление факта искажения кодового слова выполняется умножением полученного искаженного слова на проверочную матрицу кода. Результатом такого умножения будет вектор, называемый синдромом кодового слова.

Выполним такое умножение для наших исходных (7-го вектора с ошибкой) данных.

$$\left(e_{\langle 7\rangle}+u_{\langle 7\rangle}\right)\times H^{t}_{[n-k,k]}=\langle 0101100\rangle\times\begin{bmatrix}011\\101\\110\\111\\100\\010\\001\end{bmatrix}=\langle 110\rangle=S_{\langle n-k\rangle}.$$

В) для ***13***-го исходного слова внесем ту же самую ошибку и вычислим синдром слова $u_{<7>}+e_{<7>}=<\underbrace{1101001}_{13-е слово\ кода}>+<\underbrace{0010000}_{слово\ ошибки}>=<\underbrace{1111001}_{13-е\ слово\ с\ ошибкой}>$, выполним умножение

$$\left(e_{\langle 7\rangle}+u_{\langle 7\rangle}\right)\times H^{t}_{[n-k,k]}=\langle 1111001\rangle\times\begin{bmatrix}011\\101\\110\\111\\100\\010\\001\end{bmatrix}=\langle 110\rangle=S_{\langle n-k\rangle},$$

<1 1 1 1 0 0 1>	<1 1 1 1 0 0 1>	<1 1 1 1 0 0 1>
<0 1 1 1 1 0 0>	<1 0 1 1 0 1 0>	<1 1 0 1 0 0 1>
0 1 1 1 0 0 0=1	1 0 1 1 0 0 0=1	1 1 0 1 0 0 1=0.

В результате такого умножения на приемном конце канала получили вектор-синдром S<n-k>, размерность которого (n–k). Если синдром S<3>= <0,0,0> нулевой, то делается вывод о том, что принятое на приемной стороне слово принадлежит коду С и передано без искажений. Если синдром не равен нулю S<3> ≠ <0,0,0>, то его значение указывает на наличие ошибки и ее место в слове. Искаженный разряд соответствует номеру позиции столбца матрицы H[n-k, n], совпадающего с синдромом. После этого искаженный разряд исправляется, и полученное слово обрабатывается декодером далее. На практике для каждого принятого слова сразу вычисляется синдром и при наличии ошибки, она автоматически устраняется.

Итак, при вычислениях получен синдром S=<110> для обоих слов одинаковый. Смотрим на проверочную матрицу и отыскиваем в ней столбец, совпадающий с синдромом. Это третий слева столбец. Следовательно, ошибка допущена в третьем слева разряде, что совпадает с условиями примера. Этот третий разряд изменяется на противоположное значение и мы вернули принятые декодером слова к виду кодовых. Ошибка обнаружена и исправлена.

Вот собственно и все, именно так устроен и работает классический (7, 4)-код Хемминга.

3. ИНДИВИДУАЛЬНОЕ ЗАДАНИЕ

1.На основе информационного сообщения, представленного символами русского/английского алфавитов, служебными символами и цифрами, содержащегося в некотором текстовом файле (согласовать с преподавателем), сформировать информационное сообщение в двоичном виде; длина сообщения в бинарном виде должна быть не менее 16 символов. Для выполнения этого задания можно использовать коды ASCII символов алфавита.

2.Для полученного информационного слова построить проверочную матрицу Хемминга (значение минимального кодового расстояния согласовать с преподавателем).

3.Используя построенную матрицу, вычислить избыточные символы (слово).

4.Принять исходное слово со следующим числом ошибок: 0, 1, 2. Позиция ошибки определяется (генерируется) случайным образом.

5.Для полученного слова используя уже известную проверочную матрицу Хемминга, вновь вычислить избыточные символы (обозначим их), используя выражение

$$S = H \cdot (Y_n)^T = Y_n \cdot H^T,$$

6.Вычислить и проанализировать синдром. В случае, если анализ синдрома показал, что информационное сообщение было передано с ошибкой (или 2 ошибками), сгенерировать унарный вектор ошибки , –

вектор ошибки и исправить одиночную ошибку, используя формулу

$$Y_n = X_n + E_n,$$

проанализировать ситуацию при возникновении ошибки в 2 битах.

7.Результаты оформить в виде отчета. Файл с анализируемой информацией (а соответственно и интерфейс приложения) должен содержать исходное информационное сообщение, значения величин *k, r, n,* проверочную матрицу Хемминга , слово , , , , , синдром *S,* вектор ошибки . Программа не должна быть чувствительна к длине информационного сообщения.

Некоторой сложностью, с которой можно столкнуться при выполнении данной лабораторной работы, является выбор и реализация алгоритма построения проверочной матрицы Хемминга. Существует несколько способов, позволяющих заполнить матрицу Хемминга. Одним из простых является следующий. Поскольку известно, что минимальный вес столбцов в подматрице *P'* равен 2, вычисляем значение бинома Ньютона по следующей формуле:

$$C_{wt}^{r} = \frac{r!}{wt(r-wt)!},$$

где $wt = 2$, а $r = \log_2 k + 1$; wt – это вес (количество единиц).

Бином Ньютона позволяет вычислить количество различных комбинаций из нулей и единиц, которые мы можем получить при заданных значениях *r* и *wt.* Ниже приведен синтаксис функций, позволяющих вычислить бином Ньютона

```
int fact (int n) //функция вычисления факториала натурального числа n
{
int answer; //переменная, которая будет хранить
//значение факториала натурального числа n
if (n==1)return 1; //1 (1!=1)
//рекурсивное обращение функции fact() к самой себе
answer=fact(n - 1)*n;
return (answer);//возвращение результата работы функции в место
//вызова ее в теле основной программы
}
//функция вычисления бинома Ньютона
int binom(int wt,int r) {
int C; //переменная, которая будет хранить значение бинома Ньютона
C=fact(r)/(fact(wt)*fact(r - wt));
return (C); //возвращение результата работы
//функции в место вызова ее в теле основной программы
}
```

Листинг Пример реализации алгоритма для вычисления бинома Ньютона (числа сочетаний)

Далее переводя натуральные числа, начиная с 3 (первое натуральное число, которое при переводе в двоичную систему счисления имеет вес, равный 2) в двоичную систему счисления и дополняя полученную комбинацию нулями до *r* позиций (если это необходимо), по очереди заполняем столбцы подматрицы *P'*.

Например, положим, что $r = 5$, $wt = 2$.

Тогда

$$C_2^5 = \frac{5!}{2(5-2)!} = 10 .$$

$3_2 = 11$, т. е. первым столбцом матрицы *P'* будет вектор [00011]

дополнено в соответствии со значением *r*

$4_2 = 100$ – wt $(4_2) = 1$ – не подходит,

$5_2 = 101$ – wt $(5_2) = 2$.

Следовательно, второй столбец матрицы $A_{k,r}$: 00101, и т. д.

Программно это может выглядеть в соответствии с листингом

```
itoa(num,buf,2); //представление натурального
//числа в двоичной системе счисления
int len=strlen(buf);
//переменная len хранит длину
//полученной двоичной последовательности
sum=0; //переменная, используемая для вычисления
//веса полученной двоичной последовательности
//(двоичного представления натурального числа n)
for (int i=0;i<len;i++)
{
if (buf[i]=='1')
sum++; //при встрече в двоичном
//представлении числа символа '1' счетчик sum
//увеличивается на 1
}
if (sum==wt) //если вес числа в двоичном
//представлении равен текущему
//весу (первоначально wt=2)
{
count++; //тогда число комбинаций,
//подходящих для заполнения
//столбцов матрицы Ak,r,
//увеличивается на 1
if (len<r) //если длина полученной
//двоичной последовательности
//меньше r, происходит
//увеличение ее на (r - длина
//последовательности) нулей
{
for (int j=0;j<(r - len);j++)
{
char buf1[80]={'0'};
дополнено в соответствии со значением r
```

```
strcat(buf1,buf);
strcpy(buf,buf1);
}
}
for (int i=0;i<r;i++){
//заполнение столбца матрицы Ak,r
A[i][count - 1]= buf[i];}
}
num++; // переход к следующему натуральному числу
```

Листинг Программная реализация алгоритма создания проверочной матрицы кода Хемминга с $d_{min} = 3$

В случае если все комбинации при заданных r и wt уже исчерпаны, увеличиваем вес wt столбцов матрицы на единицу и опять вычисляем бином Ньютона.

В заполнении матрицы I нет никаких сложностей. Для начала элементы матрицы инициализируются нулями. Если общее количество столбцов матрицы $H_{n,r}$ $n = k + r$, а первый столбец матрицы I имеет индекс k (нумерация элементов матрицы $H_{n,r}$ начинается с 0), тогда это можно реализовать с помощью кода, представленного на листинге 3

```
for (int i=k; i<n; i++)
{
 for (int j=0;j<r;j++)
 {
   if (j=i - k) I[i,j]=1;
   else I[i,j]=0;
 }
}
```

Листинг 3. Фрагмент кода для заполнения столбцами единичной матрицы

Далее при вычислении i-го символа слова X_r достаточно подсчитать количество пар (1; 1) в соответствующих позициях слова X_k и i-й строки подматрицы P'. Если количество таких пар четное, т. е. по модулю 2 равно 0, то i-й элемент слова X_r[i] равен 0, в обратном случае – 1.

Вычисление синдрома – это сложение по модулю 2 векторов (слов) Y_r и Y_r'. Пусть размерность этих векторов равна r (нумерацию элементов векторов примем с 0). Тогда можно использовать код, показанный на листинге 4.

```
for (int i=0;i<r;i++)
{
  if (Yr[i]<>Yr'[i])
    s[i]=1;
  else s[i]=0;
}
```

Листинг 4. Фрагмент кода для вычисления синдрома

Из последнего следует, что i-й элемент синдрома S будет равен 0, если i-е элементы векторов Y_r и Y_r' равны между собой, и 1 –в противном случае. Если анализ синдрома показал, что переданное слово содержит ошибку,

ищем номер позиции в слове Y_n, в которой эта ошибка возникла. Зная позицию ошибки, можем сформировать унарный вектор ошибки E_n.
Пусть позицию ошибки хранит некоторая переменная mist.
Первоначально необходимо проинициализировать все элементы вектора E_n нулями, далее выполнить анализ и вычисления в соответствии с листингом

```
for (int i=0;i<n;i++)
{
  if (i<>mist) E[i]=0;
  else E[i]=1;
}
```

Листинг Фрагмент кода программы для определения вектора ошибки

Чтобы исправить ошибку, необходимо сложить по модулю 2
векторы Y_n и E_n в соответствии с выражением

$$Y_n = X_n + E_n,$$

где $E_n = e_1, e_2, \ldots, e_n$ – вектор ошибки.

ВОПРОСЫ ДЛЯ КОНТРОЛЯ

1. В чем заключается цель и функциональная сущность преобразования информации на основе избыточного кодирования?
2. Пояснить зависимость r от длины информационного слова k. Охарактеризовать относительную избыточность сообщения и время его передачи по сети.
3. Записать проверочную матрицу кода Хемминга с $d_{min} = 3$ и $d_{min} = 4$ для $k = 4; 6; 8; 9; 10; 15; 16$.
4. Записать проверочную матрицу кода простой четности для k из вопроса 3. Пояснить на примере определение минимального кодового расстояния Хемминга для данного кода.
5. Предположим, есть выбор (при построении матрицы кода) между вектор-столбцами большего и меньшего веса. Какой вариант Вы предпочтете и почему?
6. Какие коды относятся к помехоустойчивым? Какими общими свойствами они характеризуются?
7. Для чего в помехоустойчивые коды вводится избыточность?
8. Какие существуют классы *помехоустойчивых* кодов?
9. Какие коды относятся к *блочным помехоустойчивым кодам*? В каких случаях их целесообразно использовать?
10. Как определяются операции сложения и умножения в поле двоичных символов *GF(2)* (*операции сложения и умножения по модулю 2*)?

11. Какие коды называются *линейными блочными кодами*? Какие коды обладают свойством *систематичности*?
12. В чем заключается *кодирование с проверкой на четность*? Какова избыточность такого кода? В чем достоинства и недостатки этого кода?
13. Какой канал передачи информации описывается моделью *двоичного симметричного канала*?
14. В чем заключается процедура обнаружения и исправления ошибок *итеративным кодом*? Каковы достоинства и недостатки данного кода?
15. Какие существуют способы задания *линейных блочных кодов*? Из каких основных частей строится *кодовое слово* линейного блочного систематического кода?
16. Что такое *система проверочных уравнений* линейного блочного кода?
17. Что такое *порождающая матрица* линейного блочного кода? Каковы ее свойства? Какова структура порождающей матрицы?
18. Как, используя порождающую матрицу, построить систему проверочных уравнений линейного блочного кода?
19. Что такое *проверочная матрица* линейного блочного кода? Каковы ее свойства?
20. Какова структура проверочной матрицы линейного блочного кода? Какая часть проверочной матрицы соответствует информационным символам, а какая – проверочным?
21. Как, используя проверочную матрицу, построить систему проверочных уравнений линейного блочного кода?
22. Как описывается *вектор ошибок* в двоичном канале связи? В чем заключается задача декодирования переданного кодового слова?
23. Что такое *кодовый синдром* линейного блочного кода? Как он определяется?
24. Каким свойством характеризуется *синдром* принятого вектора? В каких случаях кодовый синдром не позволяет обнаружить ошибки в переданной последовательности?
25. Как с помощью кодового синдрома *обнаруживаются* и *исправляются* ошибки линейным блочным кодом?
26. Как определяются *вес* и *расстояние Хэмминга* для двоичных последовательностей?
27. Что такое *минимальное кодовое расстояние Хэмминга* линейного блочного кода? Как оно определяется?
28. Каково необходимое и достаточное условие *обнаружения* линейным блочным кодом ошибок заданной кратности?
29. Каково необходимое и достаточное условие *исправления* линейным блочным кодом ошибок заданной кратности?
30. Каковы *необходимое* и *достаточное условия* существования помехоустойчивого кода?
31. Как определяется *минимальное количество проверочных символов* для линейного блочного кода с заданными характеристиками?

32. Как построить порождающую матрицу линейного блочного кода с заданными характеристиками?
33. Какие линейные блочные коды называются *кодами Хэмминга*?
34. Как определяется количество информационных и проверочных символов для кода *Хэмминга*?
35. Как строятся кодовые слова кода *Хэмминга*?
36. Как составляется проверочная матрица двоичного кода *Хэмминга*?
37. Чему соответствует значение синдрома при использовании кода *Хэмминга*?
38. Как происходит декодирование кода *Хэмминга*?
39. Как строится порождающая матрица кода *Хэмминга*?