**Практическая работа 2**

**КОДЫ ХЕММИНГА**

**Цель работы** – исследование помехоустойчивости кодов Хемминга при передаче цифровых визуальных данных.

**Теоретическая часть**

При передаче информации по каналу связи с помехами в принятых данных могут возникать ошибки. Если такие ошибки имеют небольшую величину или возникают достаточно редко, информация может быть использована потребителем. При большом числе ошибок полученной информацией пользоваться нельзя.

Для уменьшения количества ошибок, возникающих при передаче информации по каналу с помехами, может быть использовано к*одирование в канале*, или *помехоустойчивое кодирование*.

Возможность использования кодирования для уменьшения числа ошибок в канале была теоретически показана К. Шенноном в 1948 году в его работе "Математическая теория связи". В ней было сделано утверждение, что *если скорость создания источником сообщений (производительность источника) не превосходит некоторой величины, называемой пропускной способностью канала, то при соответствующем кодировании и декодировании можно свести вероятность ошибок в канале к нулю*.

Кодирование с исправлением ошибок представляет собой *метод обработки сообщений, предназначенный для повышения надежности передачи по цифровым каналам.* Хотя различные схемы кодирования очень не похожи друг на друга и основаны на различных математических теориях, всем им присущи два общих свойства.

Первое − *использование избыточности*. Закодированные последовательности всегда содержат дополнительные, или избыточные, символы. *Количество символов в кодовой последовательности* ***Y*** *всегда больше, чем необходимо для однозначного представления любого сообщения* ***λi*** *из алфавита.*

Второе − *свойство усреднения*, означающее, что *избыточные символы зависят от нескольких информационных символов*, то есть информация, содержащаяся в кодовой последовательности ***X,*** перераспределяется также и на избыточные символы.

Существует два больших класса корректирующих кодов − *блочные и сверточные.* Определяющее различие между этими кодами состоит в отсутствии или наличии памяти кодера.

*Кодер для блочных кодов* делит непрерывную информационную последовательность ***X*** на блоки-сообщения длиной ***m*** символов.

*Кодер канала* преобразует блоки-сообщения ***X*** в более длинные двоичные последовательности ***Y***, состоящие из ***n*** символов и называемые *кодовыми словами.* Символы (***n-m***), добавляемые к каждому блоку-сообщению кодером, называются *избыточными*. Они не несут никакой дополнительной информации, и *их функция состоит в обеспечении возможности обнаруживать (или исправлять) ошибки*, возникающие в процессе передачи.

Очевидно, что ***m***-разрядным двоичным словом можно представить ***2m*** возможных значений из алфавита источника и им соответствует ***2m*** кодовых слов на выходе кодера. *Такое множество* ***2m*** *кодовых слов называется* ***блочным кодом****.*

Блочный код длиной ***n*** символов, состоящий из ***2m*** кодовых слов, называется линейным (***n, m***)-кодом при условии, что все его ***2m*** кодовых слов образуют ***m***-мерное подпространство векторного пространства ***n-*** последовательностей двоичного поля ***GF(2)***.

*Полем* называется множество математических объектов, которые можно складывать, вычитать, умножать и делить. *Алфавит из двух символов 0 и 1 вместе со сложением и умножением по* ***mod2*** *называется полем из двух элементов и обозначается как* ***GF(2)***. К полю ***GF(2)*** применимы все методы линейной алгебры, в том числе матричные операции. Отметим, что все действия над символами в двоичных кодах выполняются по модулю 2.

Систематический код имеет формат, изображенный на рис. 2.1, то есть содержит *неизменную информационную часть длиной* ***m*** *символов* и *избыточную (проверочную) длиной* ***k*** *=* ***n – m***  *символов*.

|  |  |
| --- | --- |
| ***n*** | |
| ***m*** | ***k*** |

Рис. 2.1.

Блочный код, обладающий свойствами линейности и систематичности, *называется линейным блочным систематическим (****n, m****)-кодом.*

*Коды Хемминга*.

Коды Хэмминга являются самоконтролирующимися кодами, то есть кодами, позволяющими автоматически обнаруживать ошибки при передаче данных. Для их построения достаточно приписать к каждому слову один добавочный (контрольный) двоичный разряд и выбрать цифру этого разряда так, чтобы общее количество единиц в изображении любого числа было, например, четным.

Одиночная ошибка в каком-либо разряде передаваемого слова (в том числе, может быть, и в контрольном разряде) изменит четность общего количества единиц. Счетчики по модулю 2, подсчитывающие количество единиц, которые содержатся среди двоичных цифр числа, могут давать сигнал о наличии ошибок.

При этом невозможно узнать, в каком именно разряде произошла ошибка, и, следовательно, нет возможности исправить её. Остаются незамеченными также ошибки, возникающие одновременно в двух, четырёх, и т.д. — в четном количестве разрядов. Впрочем, двойные, а тем более четырёхкратные ошибки полагаются маловероятными.

Коды, в которых возможно автоматическое исправление ошибок, называются самокорректирующимися. Для построения самокорректирующегося кода, рассчитанного на исправление одиночных ошибок, одного контрольного разряда недостаточно. Количество контрольных разрядов ***k*** должно быть выбрано так, чтобы удовлетворялось неравенство

***2k*** ≥ ***k*** + ***m*** +1 или ***k*** ≥ log2(***k*** + ***m*** + 1),

где ***m*** — количество основных двоичных разрядов кодового слова. Минимальные значения ***k*** при заданных значениях ***m***, найденные в соответствии с этим неравенством, приведены в таблице 2.1.

Таблица 2.1

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ***m*** | 2-4 | 5-11 | 12-26 | 27-57 | 58-120 | 121-247 | 248-502 | 503-1013 |
| ***k***min | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |

Основными характеристиками самокорректирующихся кодов являются:

1. Число разрешенных и запрещенных комбинаций. Если ***n*** - число символов в блоке, ***k*** - число проверочных символов в блоке, ***m*** - число информационных символов, то ***2n*** - число возможных кодовых комбинаций, ***2m*** - число разрешенных кодовых комбинаций, (***2n*** ***– 2m***) - число запрещенных комбинаций.

2. Скорость кода и избыточность. Скорость кода определяется отношением ***m***/***n***. Величина (***n***/***m*** ***–***1) ***⋅***100% определяет его избыточность в процентах.

3. Минимальное кодовое расстояние. Минимальным кодовым расстоянием ***dmin*** называется минимальное число искаженных символов, необходимое для перехода одной разрешенной комбинации в другую.

4. Число обнаруживаемых и исправляемых ошибок. Если ***g*** - количество ошибок, которое код способен исправить, то необходимо и достаточно, чтобы выполнялось условие ***dmin*** ≥ 2***g*** + 1. Следовательно, зная значение ***d***, мы можем определить количество исправляемых ошибок в блоке ***g*** = ***INT***[(***dmin*** -1)/2].

Построение кодов Хемминга основано на принципе проверки на четность числа единичных символов: к последовательности добавляется такой элемент, чтобы число единичных символов в получившейся последовательности было четным. Для каждого числа проверочных символов ***k*** существует классический код Хемминга с маркировкой (***n***, ***k***): (7, 4); (15,11); (31,26); (63,57)….

Наиболее удобным и наглядным способом описания линейных блочных кодов является их задание с использованием *порождающей матрицы*, являющейся компактной формой представления системы проверочных уравнений:

|  |  |  |
| --- | --- | --- |
|  | ***1 0 0 … 0*** | ***P00 P01 . . . . P0, n- k- 1*** |
| ***G =*** | ***0 1 0 … 0*** | ***P10 P11 . . . . P1, n- k- 1*** |
|  | ***………*** | ***………………*** |
|  | ***0 0 0 … 1*** | ***Pk- 1, 0 Pk- 1, 1 . . . . Pk- 1, n- k- 1*** |
|  | единичная  матрица **I**  k\*k | матрица **Р**  k\*(n- k) |

***Определение.*** *Линейный блочный систематический (****n,k****)-код полностью определяется матрицей* ***G*** *размером* ***k \* n*** *с двоичными матричными элементами. При этом каждое кодовое слово является линейной комбинацией строк матрицы* ***G****, а каждая линейная комбинация строк* ***G***  *- кодовым словом.*

Пусть ***m*** = (***m0 , m1 , . . . , mk -1***) будет тем блоком-сообщением, который необходимо закодировать с использованием данного кода.

Тогда соответствующим ему кодовым словом ***U*** будет

***U*** = ***m***⋅ ***G***.

С учетом структуры матрицы ***G*** символы кодового слова ***U*** будут такими:

*для* ***i = 0, 1, 2,. . . , k- 1***

***Ui = mi***

*для* ***i = k, k+1,. . . , n***

***Ui = m0⋅ P0j + m1⋅ P1j + m2⋅ P2j +…+ mk- 1⋅ Pk- 1, j***

Иными словами, ***k*** *крайних левых символов кодового слова совпадают с символами кодируемой информационной последовательности*, а остальные (***n - к***) *символов являются* *линейными комбинациями символов информационной последовательности*.

Определенный таким образом код называется *линейным блочным систематическим (****n,k****)-кодом с обобщенными проверками на четность, а* з*адающая его матрица* ***G***  *называется порождающей матрицей кода.*

В качестве примера рассмотрим известный (***7,4***)-код Хемминга, являющийся классической иллюстрацией простейших кодов с исправлением ошибок.

Пусть ***m*** = (***m0, m1, m2, m3***) будет тем сообщением, или информационной последовательностью, которую нужно закодировать.

Порождающая матрица ***G*** для (***7. 4***)-кода Хемминга имеет вид

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | ***1*** | ***0*** | ***0*** | ***0*** | ***1*** | ***1*** | ***0*** |
| ***G(7,4) =*** | ***0*** | ***1*** | ***0*** | ***0*** | ***0*** | ***1*** | ***1*** |
|  | ***0*** | ***0*** | ***1*** | ***0*** | ***1*** | ***1*** | ***1*** |
|  | ***0*** | ***0*** | ***0*** | ***1*** | ***1*** | ***0*** | ***1*** |

Тогда символы соответствующего кодового слова определяются следующим образом:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | ***1*** | ***0*** | ***0*** | ***0*** | ***1*** | ***1*** | ***0*** |  |
| ***U = m⋅ G*** ***= ( m0 m1 m2 m3 )*** | ***0*** | ***1*** | ***0*** | ***0*** | ***0*** | ***1*** | ***1*** | ***=*** |
|  | ***0*** | ***0*** | ***1*** | ***0*** | ***1*** | ***1*** | ***1*** |  |
|  | ***0*** | ***0*** | ***0*** | ***1*** | ***1*** | ***0*** | ***1*** |  |

***= (m0 , m1 , m2 , m3 , m0 + m2 + m3 , m0 + m1 + m2 , m1 + m2 + m3 ),***

или

***U0 = m0 , U1 = m1 , U2 = m2 , U3 = m3 ,***

***U4 = m0 + m2 + m3 , U5 = m0 + m1 + m2 , U6 = m1 + m2 + m3***

Например, пусть  ***m*** **=** (***1 0 1 1***), тогда соответствующее кодовое слово будет иметь вид ***U***  = **(*1 0 1 1 1 0 0*)**.Или другой пример: пусть ***m*** **=** (***1 0 0 0***) тогда ***U*** = **(*1 0 0 0 1 1 0*)**.

В соответствии с приведенным выше определением строки матрицы ***G*** сами являются кодовыми словами данного кода, а все остальные кодовые слова - линейными комбинациями строк порождающей матрицы.

На основании порождающей матрицы ***G(7,4)*** или приведенной системы проверочных уравнений легко реализовать схему кодирования для рассматриваемого (***7,4***)-кода Хемминга (рис. 2.2).

Кодер работает точно так же, как и при простой проверке на четность, но теперь выполняет не одну общую, а несколько частичных проверок, формируя, соответственно, несколько проверочных символов.

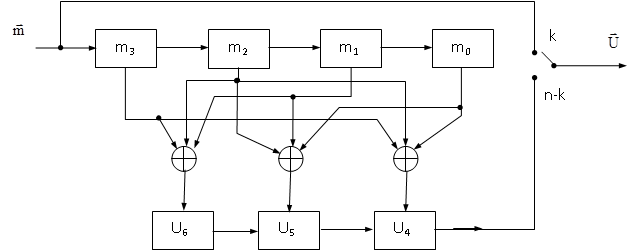


Рис. 2.2

Прежде чем говорить об обнаружении и исправлении ошибок корректирующими кодами, определим само понятие ошибки и методы их описания.

Пусть ***U = (U0 , U1 ,… Un )*** является кодовым словом, переданным по каналу с помехами, а ***r = (r0 , r1 , ... rn)*** - принятой последовательностью, возможно, отличающейся от переданного кодового слова ***U***. Отличие ***r*** от ***U*** состоит в том, что некоторые символы ***ri*** принятой последовательности могут отличаться от соответствующих символов ***Ui*** переданного кодового слова.

Например, ***U =* (*0 0 0 1 0 0 0*),** а ***r =* (*0 0 0 0 0 0 0*)**, то есть произошла ошибка в четвертом символе кодового слова, **1** перешла в **0** . Или другой пример: передано кодовое слово ***U =* (*0 0 1 1 1 1*),** а принятая последовательность имеет вид ***r =* (*1 0 1 1 1 1 1*),** то есть ошибка возникла в первом бите кодового слова, при этом **0** перешел в единицу.

Для описания возникающих в канале ошибок используют вектор ошибки, обычно обозначаемый как ***e*** и представляющий собой двоичную последовательность длиной ***n*** с единицами в тех позициях, в которых произошли ошибки.

Так, вектор ошибки ***e =* (*0 0 0 1 0 0 0* )** означает однократную ошибку в четвертой позиции (четвертом бите), вектор ошибки ***e =* (*1 1 0 0 0 0 0*)** - двойную ошибку в первом и втором битах и т.д.

Тогда при передаче кодового слова ***U*** по каналу с ошибками принятая последовательность ***r*** будет иметь вид

***r*** = ***U + e*** ,

Приняв вектор ***r***, декодер сначала должен определить, имеются ли в принятой последовательности ошибки. Если ошибки есть, то он должен выполнить действия по их исправлению.

Чтобы проверить, является ли принятый вектор кодовым словом, декодер вычисляет (***n-k***)-последовательность, определяемую следующим образом:

***S*** = (***S0***, ***S1***, … , ***Sn-k-1***) = ***r*** ⋅ ***HT***,

где ***HT*** — проверочная транспонированная матрица. Проверочная матрица определяется из порождающей матрицы ***G***

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | ***P00*** | ***P10*** | ***…*** | ***Pk-1, 0*** | ***1*** | ***0*** | ***0*** | ***…*** | ***0*** |  |
|  | ***P01*** | ***P11*** | ***…*** | ***Pk-1, 1*** | ***0*** | ***1*** | ***0*** | ***…*** | ***0*** |  |
| ***H =*** | ***P22*** | ***P12*** | ***…*** | ***Pk-1, 2*** | ***0*** | ***0*** | ***1*** | ***…*** | ***0*** | , |
|  | ***…*** | ***…*** | ***…*** | ***…*** | ***…*** | ***…*** | ***…*** | ***…*** | ***…*** |  |
|  | ***P0, n-k-1*** | ***P1, n-k-1*** | ***…*** | ***Pk-1, n-k-1*** | ***0*** | ***0*** | ***0*** | ***…*** | ***1*** |  |
|  |  |  |  |  |  |  |  |  |  |  |
|  | ***PT*** | | | | ***I1(n-k)×(n-k)*** | | | | |  |

где ***PT*** - транспонированная подматрица ***P*** из порождающей матрицы ***G ;***

***I1(n-k)×(n-k)***  - единичная матрица соответствующего размера. Видно, что единичная и проверочная подматрицы в ***G*** и ***H*** поменялись местами, кроме того, изменился их размер.

Для рассматриваемого примера (***7,4***)-кода Хемминга проверочная матрица ***H*** имеет вид

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | ***1*** | ***0*** | ***1*** | ***1*** | ***1*** | ***0*** | ***0*** |
| ***H(7,4)=*** | ***1*** | ***1*** | ***1*** | ***0*** | ***0*** | ***1*** | ***0*** |
|  | ***0*** | ***1*** | ***1*** | ***1*** | ***0*** | ***0*** | ***1*** |

Проверочная матрица позволяет легко определить, *является ли принятая последовательность кодовым словом данного кода*.

Пусть, к примеру, принята последовательность символов ***r*** = (***1011001***), тогда

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | ***1*** | ***0*** | ***1*** | ***1*** | ***1*** | ***0*** | ***0*** | ***T*** |
| ***r\* HT = (1011001)*** | ***1*** | ***1*** | ***1*** | ***0*** | ***0*** | ***1*** | ***0*** | ***= (1 1 0) ≠ 0 .*** |
|  | ***0*** | ***1*** | ***1*** | ***1*** | ***0*** | ***0*** | ***1*** |  |

Отсюда можно сделать вывод, что последовательность ***r*** = (**1011001**) *не является кодовым словом данного кода.*

Рассмотрим другой пример. Допустим, принята последовательность   
***r*** = (**0010111**), тогда

|  | ***1*** | ***0*** | ***1*** | ***1*** | ***1*** | ***0*** | ***0*** | ***T*** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ***r ⋅ HT = (0010111)*** | ***1*** | ***1*** | ***1*** | ***0*** | ***0*** | ***1*** | ***0*** | ***= (0 0 0) = 0 ,*** |
|  | ***0*** | ***1*** | ***1*** | ***1*** | ***0*** | ***0*** | ***1*** |  |

то есть двоичная последовательность ***r*** *принадлежит коду с проверочной матрицей* ***H.***

Таким образом, ***r*** *является кодовым словом тогда, и только тогда, когда*   
***S*** ***=*** ***(00..0),*** *и не является кодовым словом данного кода, если* ***S ≠ 0.*** Следовательно, ***S*** используется для обнаружения ошибок, *ненулевое значение* ***S*** *служит признаком наличия ошибок в принятой последовательности*. *Поэтому вектор* ***S*** *называется синдромом принятого вектора* ***r****.*

Некоторые сочетания ошибок, используя синдром, обнаружить невозможно. Например, если переданное кодовое слово ***U*** под влиянием помех превратилось в другое действительное кодовое слово ***V*** этого же кода, то синдром

***S*** = ***V*** \* ***HT*** = **0 .**

В этом случае декодер ошибки не обнаружит и, естественно, не попытается ее исправить.

Сочетания ошибок такого типа называются не обнаруживаемыми. При построении кодов необходимо стремиться к тому, чтобы они обнаруживали наиболее вероятные сочетания ошибок.

Для рассматриваемого в качестве примера линейного блочного систематического (***7,4***)-кода Хемминга синдром определяется следующим образом: пусть принят вектор ***r*** = ( ***r0 , r1 , r2 , r3 , r4 , r5 , r6***), тогда

|  |  |  |
| --- | --- | --- |
|  | ***1 0 1 1 1 0 0*** | ***T*** |
| ***S*** ***= r\* H(7,4)T = ( r0, r1, r2, r3, r4, r5, r6 )*** | ***1 1 1 0 0 1 0*** | ***=*** |
|  | ***0 1 1 1 0 0 1*** |  |

***= (r0 + r2 + r3 + r4 ), ( r0 + r1 + r2 + r5 ), ( r1 + r2 + r2 + r6 ),***

или

***S0 = r0 + r2 + r3 + r4 ,   
 S1 = r0 + r1 + r2 + r5 ,***  ***S2 = r1 + r2 + r2 + r6 .***

Основываясь на полученных соотношениях, можно легко организовать схему для вычисления синдрома. Для (***7,4***)-кода Хемминга она приведена на рис. 2.3.

Покажем, как можно использовать синдром принятого вектора не только для обнаружения, но и для исправления ошибок.

Пусть ***U*** = ( ***U0 , U1 , …, Un-1*** ), ***e*** = ( ***е0 , е1, …, еn-1***) и ***r*** = ( ***r0 , r1, r2 , …, rn-***1) являются передаваемым кодовым словом, вектором-ошибкой и принятым вектором соответственно. Тогда

***r*** = ***U + e***

и синдром

***S = r⋅HT = (U + e )⋅ HT = U⋅ HT + e⋅ HT = 0 + e⋅ HT = e⋅ HT ,***

поскольку для любого кодового слова ***U ⋅ HT*** = ***0.***

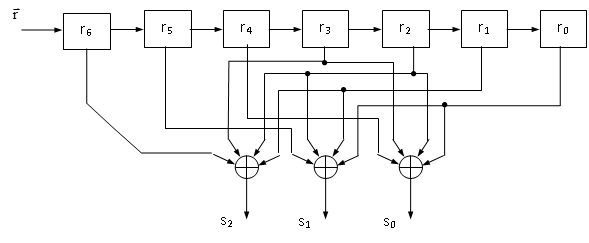


Рис. 2.3

Таким образом, синдром принятой последовательности ***r*** зависит только от ошибки, имеющей место в этой последовательности, и совершенно не зависит от переданного кодового слова. Задача декодера, используя эту зависимость, определить элементы (координаты) вектора ошибок. Найдя вектор ошибки можно восстановить кодовое слово как

***U\* = r + e*** .

На примере одиночных ошибок при кодировании с использованием линейного блочного (***7,4***)-кода покажем, как вектор ошибки связан с синдромом, и как, имея синдром, локализовать и устранить ошибки, возникшие при передаче. *Найдем значения синдрома для всех возможных одиночных ошибок в последовательности из семи символов:*

***e =*** ( ***0000000*** ),

|  |  |  |
| --- | --- | --- |
|  | ***1011100*** | ***T*** |
| ***e ⋅ HT =* (*0000000*)*⋅*** | ***1110010*** | ***=*** (***000***)***;*** |
|  | ***0111001*** |  |

***e0 =*** ( ***1000000*** ),

|  |  |  |
| --- | --- | --- |
|  | ***1011100*** | ***T*** |
| ***e0⋅ HT =* (*1000000* ) *⋅*** | ***1110010*** | *=* (***110***)***;*** |
|  | ***0111001*** |  |

***e1 =*** ( ***0100000*** ),

|  |  |  |
| --- | --- | --- |
|  | ***1011100*** | ***T*** |
| ***e1⋅ HT =* ( *0100000* ) *⋅*** | ***1110010*** | ***= (011);*** |
|  | ***0111001*** |  |

***e2 =*** ( ***0010000*** ),

|  |  |  |
| --- | --- | --- |
|  | ***1011100*** | ***T*** |
| ***e2⋅ HT =* ( *0010000* ) *⋅*** | ***1110010*** | *=* (***111***)***;*** |
|  | ***0111001*** |  |

***e3 =*** ( ***0001000*** ),

|  |  |  |
| --- | --- | --- |
|  | ***1011100*** | ***T*** |
| ***e3⋅ HT =* (*0001000*) *⋅*** | ***1110010*** | *=* (***101***)***;*** |
|  | ***0111001*** |  |

***e4 =*** ( ***0000100*** ),

|  |  |  |
| --- | --- | --- |
|  | ***1011100*** | ***T*** |
| ***e4⋅ HT =* (*0000100*** ) ***⋅*** | ***1110010*** | *=* (***100***)***;*** |
|  | ***0111001*** |  |

***e5 =*** ( ***0000010*** ),

|  |  |  |
| --- | --- | --- |
|  | ***1011100*** | ***T*** |
| ***e5⋅ HT =* (*0000010***) ***⋅*** | ***1110010*** | *=* (***010***)***;*** |
|  | ***0111001*** |  |

***e6 =*** ( ***0000001*** ),

|  |  |  |
| --- | --- | --- |
|  | ***1011100*** | ***T*** |
| ***e6⋅ HT =* (*0000001*) *⋅*** | ***1110010*** | *=* (***001***). |
|  | ***0111001*** |  |

Все возможные для (***7,4***)-кода одиночные ошибки и соответствующие им векторы синдрома приведены в табл. 2.2.

Таблица 2.2

|  |  |  |
| --- | --- | --- |
| Вектор ошибки | Синдром ошибки | Десятичный код синдрома |
| ***1000000*** | ***110*** | 6 |
| ***0100000*** | ***011*** | 3 |
| ***0010000*** | ***111*** | 7 |
| ***0001000*** | ***101*** | 5 |
| ***0000100*** | ***100*** | 4 |
| ***0000010*** | ***010*** | 2 |
| ***0000001*** | ***001*** | 1 |

Из этой таблицы видно, что существует *однозначное соответствие между сочетанием ошибок (при одиночной ошибке) и его синдромом*, то есть, зная синдром, можно однозначно определить позицию кода, в которой произошла ошибка.

Например, если синдром, вычисленный по принятому вектору, равен (***111***), это значит, что произошла одиночная ошибка в третьем символе, если  
***S*** = (***001***) – то в последнем, и т.д.

Если место ошибки определено, то устранить ее уже не представляет никакого труда.

Рассмотрим, чем определяется способность блочного кода обнаруживать и исправлять ошибки, возникшие при передаче.

Пусть ***U = (U0, U1, U2, ...Un-1)*** - двоичная последовательность длиной ***n.***

*Число единиц* (ненулевых компонент) в этой последовательности *называется* *весом Хемминга* вектора ***U*** и обозначается ***w(U).***

Например, вес Хемминга вектора ***U***= (***1001011*** ) равен четырем, для вектора ***U***= (***1111111***) величина ***w(U)*** составит 7 и т.д.

Таким образом, чем больше единиц в двоичной последовательности, тем больше ее вес Хемминга.

Далее, пусть ***U*** и ***V*** будут двоичными последовательностями длиной ***n***.

*Число разрядов, в которых эти последовательности различаются, называется расстоянием Хемминга между* ***U*** *и* ***V*** и обозначается ***d(U, V).***

Например, если ***U*** = (***1001011***), а ***V*** = ( ***0100011*** ), то ***d(U, V) =*** 3.

Задав линейный код и определив все ***2k*** его кодовых слов, можно вычислить расстояние между всеми возможными парами кодовых слов. Минимальное из них называется *минимальным кодовым расстоянием кода* и обозначается ***dmin***.

Можно проверить и убедиться, что минимальное кодовое расстояние для рассматриваемого нами в примерах (***7,4***)-кода равно трем: ***dmin(7,4) = 3***. Для этого нужно записать все кодовые слова (***7,4***)-кода Хемминга (всего 16 слов), вычислить расстояния между их всеми парами и взять наименьшее значение. Однако можно определить ***dmin*** блочного кода и более простым способом.

Доказано, что расстояние между нулевым кодовым словом и одним из кодовых слов, входящих в порождающую матрицу (строки порождающей матрицы линейного блочного кода сами являются кодовыми словами, по определению), равно ***dmin***. Но расстояние от любого кодового слова до нулевого равно весу Хемминга этого слова. Тогда ***dmin*** равно *минимальному весу Хемминга* *для всех строк порождающей матрицы кода*.

Если при передаче кодового слова по каналу связи в нем произошла *одиночная ошибка*, то расстояние Хемминга между переданным словом ***U*** и принятым вектором ***r*** будет равно единице. Если при этом одно кодовое слово не перешло в другое (а при ***dmin***> 1 и при одиночной ошибке это невозможно), то ошибка *будет обнаружена* при декодировании.

В общем случае если блочный код имеет минимальное расстояние ***dmin***, то *он может обнаруживать любые сочетания ошибок при их числе, меньшем или равном* ***dmin - 1***, поскольку никакое сочетание ошибок при их числе, меньшем, чем ***dmin - 1,*** не может перевести одно кодовое слово в другое.

Но ошибки могут иметь кратность и большую, чем ***dmin- 1***, и тогда они *останутся необнаруженными*.

При этом среднюю вероятность не обнаруживаемой ошибки можно определить следующим образом.

Пусть вероятность ошибки в канале связи равна ***Pош***. Тогда вероятность того, что при передаче последовательности длины ***n*** в ней произойдет одна ошибка, равна

***Р1 = n Pош ⋅ ( 1- Рош)n-1,***

соответственно, вероятность ***l***-кратной ошибки –

***Pl =Cnl Pошl ⋅ ( 1- Pош)n-l***,

где ***Cnl*** - число возможных комбинаций из ***n*** символов кодовой последовательности по ***l*** ошибок.

По каналу связи передаются кодовые слова с различными весами Хемминга. Положим, что ***ai***  — число слов с весом ***i*** в данном коде (всего слов в коде длиной ***n*** равно ***A*** = ***2k***).

А теперь определим, что такое *не обнаруживаемая ошибка*. Обнаружение ошибки производится путем вычисления синдрома принятой последовательности. Если принятая последовательность не является кодовым словом (тогда синдром не равен нулю), то считается, что ошибка есть. Если же синдром равен нулю, то полагаем, что ошибки нет (принятая последовательность является кодовым словом). Возникает вопрос, то ли кодовое слово передавалось? Или же в результате действия ошибок переданное кодовое слово перешло в другое кодовое слово данного кода:

***r = U + е = V,***

то есть сумма переданного кодового слова ***U*** и вектора ошибки ***е*** даст новое кодовое слово ***V***? В этом случае, естественно, ошибка обнаружена быть не может.

Но из определения двоичного линейного кода следует, что *если сумма кодового слова и некоторого вектора* ***е*** *есть кодовое слово, то вектор* ***е*** *также представляет собой кодовое слово*. *Следовательно, необнаруживаемые ошибки будут возникать тогда, когда сочетания ошибок будут образовывать кодовые слова.*

Вероятность того, что вектор ***е*** совпадает с кодовым словом, имеющим вес ***i ,*** равна

***Pi = Pошi⋅ (1- Рош)n-i .***

Тогда полная вероятность возникновения необнаруживаемой ошибки



*Пример*: рассматриваемый нами (***7,4***)-код содержит по семь кодовых слов с весами ***w = 3*** и ***w = 4*** и одно кодовое слово с весом ***w = 7***, тогда



или, при ***Рош = 10 -3***, ***Р(Е) ≅ 7 ⋅ 10 -9.***

Другими словами, если по каналу передается информация со скоростью   
***V*** = 1кбит/с и в канале в среднем *каждую секунду* будет происходить искажение одного символа, то в среднем семь принятых слов на ***109*** переданных будут проходить через декодер без обнаружения ошибки (одна необнаруживаемая ошибка за 270 часов).

Таким образом, использование даже такого простого кода позволяет на несколько порядков снизить вероятность необнаруживаемых ошибок.

Теперь предположим, что линейный блочный код используется для исправления ошибок. Чем определяются его возможности по исправлению?

Рассмотрим пример, приведенный на рис. 2.4. Пусть ***U*** и ***V*** представляют пару кодовых слов кода *с кодовым расстоянием* ***d ,*** *равным минимальному —* ***dmin*** *для данного кода.*



Рис. 2.4.

Предположим, передано кодовое слово ***U,*** в канале произошла *одиночная ошибка* и принят вектор ***а*** (не принадлежащий коду).

Если декодирование производится оптимальным способом, то есть по методу максимального правдоподобия, то *в качестве оценки* ***U\*****нужно выбрать ближайшее к* ***а*** *кодовое слово.* Таковым в данном случае будет ***U***, следовательно, ошибка будет устранена.

Представим теперь, что произошло две ошибки и принят вектор ***b***.

Тогда при декодировании по максимуму правдоподобия *в качестве оценки будет выбрано ближайшее к* ***b*** *кодовое слово, и им будет* ***V***. Произойдет ошибка декодирования.

Продолжив рассуждения для ***dmin = 4, dmin = 5*** и т.д., нетрудно сделать вывод, что ошибки будут устранены, если их кратность ***l*** не превышает величины

***l< INT (( dmin – 1 )/2) ,***

где***INT (X)*** — целая часть ***Х***.

Так, используемый нами в качестве примера (***7,4***)-код имеет ***dmin = 3*** и, следовательно, позволяет исправлять лишь одиночные ошибки:

***l = INT (( dmin – 1 )/2)=INT((3-1)/2)=1*** .

Таким образом, *возможности линейных блочных кодов по обнаружению и исправлению ошибок определяются их минимальным кодовым расстоянием*. Чем больше ***dmin***, тем большее число ошибок в принятой последовательности можно исправить.

*А теперь определим вероятность того, что возникшая в процессе передачи ошибка не будет исправлена при декодировании.*

Пусть, как и ранее, вероятность ошибки в канале будет равна ***Рош***. Ошибки, возникающие в различных позициях кода, считаем независимыми.

Вероятность того, что принятый вектор ***r*** будет иметь какие-нибудь (одиночные, двукратные, трехкратные и т.д.) ошибки, можно определить как

***Рош  = P1 + P2 + P3 +... + Pn ,***

где***Р1*** — вероятность того, что в ***r*** присутствует одиночная ошибка;   
***Р2*** — вероятность того, что ошибка двойная и т.д.;   
***Рn*** — вероятность того, что все ***n*** символов искажены.

Определим вероятность ошибок заданной кратности:

***Р1 = п ⋅ Рош(1- Рош)n-1***; ***Р2*** *=* ***Р2ош(1- Рош) n-2***; ***Р3 = Р3ош(1- Рош)n-3*** и т.д.

Декодер, как мы показали, исправляет все ошибки, кратность которых не превышает

***,***

то есть все ошибки кратности ***J ≤ l*** будут исправлены.

Тогда ошибки декодирования - это ошибки с кратностью, большей кратности исправляемых ошибок ***l,*** и их вероятность

.

Для (***7,4***)-кода Хемминга минимальное расстояние ***dmin = 3***, т.е. ***l*** = ***1***. Следовательно, ошибки кратности 2 и более исправлены не будут и



Если ***Рош***<< ***1***, можно считать (***1- Рош***) ≈ ***1*** и, кроме того, ***Р3ош<< Р2ош***. Тогда



Так, например, при вероятности ошибки в канале ***Рош = 10 -3*** вероятность неисправления ошибки  ***Р(N) ≈ 2 ⋅10 -5***, то есть при такой вероятности ошибок в канале кодирование (***7.4***)-кодом позволяет снизить вероятность оставшихся неисправленными ошибок примерно *в пятьдесят раз*.

Если же вероятность ошибки в канале будет в сто раз меньше ***Рош = 10 -5***, то вероятность ее неисправления составит уже ***Р(N) ≈ 2⋅10 -9***, или *в 5000 раз меньше!*

Таким образом, *выигрыш от помехоустойчивого кодирования (который можно определить как отношение числа ошибок в канале к числу оставшихся неисправленными ошибок) существенно зависит от свойств канала связи.*

Если вероятность ошибок в канале велика, то есть канал не очень хороший, ожидать большого эффекта от кодирования не приходится, если же вероятность ошибок в канале мала, то корректирующее кодирование уменьшает ее в значительно большей степени.

Другими словами, *помехоустойчивое кодирование существенно улучшает свойства хороших каналов, в плохих же каналах оно большого эффекта не дает.*

*Алгоритм кодирования*

Предположим, что нужно сгенерировать код Хемминга (15, 11) для некоторого информационного кодового слова. В качестве примера возьмём 11-битовое кодовое слово ***m0*** … ***m10*** , хотя алгоритм пригоден для кодовых слов любой длины. В приведённой ниже таблице в первой строке даны номера позиций в кодовом слове, во второй — условное обозначение битов, в третьей — значения битов.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |
| ***m0*** | ***m1*** | ***m2*** | ***m3*** | ***m4*** | ***m5*** | ***m6*** | ***m7*** | ***m8*** | ***m9*** | ***m10*** |
| 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 1 |

Вставим в информационное слово контрольные биты ***k0***...*k3* таким образом, чтобы номера их позиций представляли собой целые степени двойки: 1, 2, 4, 8, 16... Получим 15-разрядное слово с 11 информационными и 4 контрольными битами. Первоначально контрольные биты устанавливаем равными нулю. На рисунке контрольные биты выделены серым цветом.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
| ***k0*** | ***k1*** | ***m0*** | ***k2*** | ***m1*** | ***m2*** | ***m3*** | ***k3*** | ***m4*** | ***m5*** | ***m6*** | ***m7*** | ***m8*** | ***m9*** | ***m10*** |
| 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 1 |

В общем случае количество контрольных бит в кодовом слове равно двоичному логарифму числа бит кодового слова (включая контрольные биты), округлённому в большую сторону до ближайшего целого. Например, информационное слово длиной 1 или 2 бита требует двух контрольных разрядов, 3- или 4-битовое информационное слово — трёх, 5...11-битовое — четырёх, 12...26-битовое — пяти и т.д.

Добавим к таблице 4 строки (по количеству контрольных битов), в которые поместим матрицу преобразования.

Каждая строка будет соответствовать одному контрольному биту (нулевой контрольный бит — верхняя строка, четвёртый — нижняя), каждый столбец — одному биту кодируемого слова. В каждом столбце матрицы преобразования поместим двоичный номер этого столбца, причём порядок следования битов будет обратный — младший бит расположим в верхней строке, старший — в нижней. Например, в третьем столбце матрицы будут стоять числа 11000, что соответствует двоичной записи числа три: 00011.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |  |
| ***k0*** | ***k1*** | ***m0*** | ***k2*** | ***m1*** | ***m2*** | ***m3*** | ***k3*** | ***m4*** | ***m5*** | ***m6*** | ***m7*** | ***m8*** | ***m9*** | ***m10*** |  |
| 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 1 |  |
| 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | ***k0*** |
| 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | ***k1*** |
| 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | ***k2*** |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | ***k3*** |

В правой части таблицы мы оставили пустым один столбец, в который поместим результаты вычислений контрольных битов. Вычисление контрольных битов производим следующим образом. Берём одну из строк матрицы преобразования (например, ***k0***) и находим её скалярное произведение с кодовым словом, то есть перемножаем соответствующие биты обеих строк и находим сумму произведений. Если произведение получилось больше единицы, находим остаток от его деления на 2. Иными словами, мы подсчитываем сколько раз в кодовом слове и соответствующей строке матрицы в одинаковых позициях стоят единицы, и берём это число по модулю 2.

Если описывать этот процесс в терминах матричной алгебры, то операция представляет собой перемножение матрицы преобразования на матрицу-столбец кодового слова, в результате чего получается матрица-столбец контрольных разрядов, которые нужно взять по модулю 2.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |  |
| ***k0*** | ***k1*** | ***m0*** | ***k2*** | ***m1*** | ***m2*** | ***m3*** | ***k3*** | ***m4*** | ***m5*** | ***m6*** | ***m7*** | ***m8*** | ***m9*** | ***m10*** |  |
| 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 1 |  |
| 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | ***1*** |
| 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | ***0*** |
| 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | ***0*** |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | ***0*** |

Полученные контрольные биты вставляем в кодовое слово за информационными битами. Кодирование по Хэммингу завершено. Полученное кодовое слово — 01001001011\_1000.

*Алгоритм декодирования*

Алгоритм декодирования по Хэммингу абсолютно идентичен алгоритму кодирования. Матрица преобразования соответствующей размерности умножается на матрицу-столбец кодового слова и каждый элемент полученной матрицы-столбца берётся по модулю 2. Полученная матрица-столбец получила название «матрица синдромов». Легко проверить, что кодовое слово, сформированное в соответствии с алгоритмом, описанным в предыдущем разделе, всегда даёт нулевую матрицу синдромов.

Матрица синдромов становится ненулевой, если в результате ошибки (например, при передаче слова по линии связи с шумами) один из битов исходного слова изменил своё значение. Предположим для примера, что в кодовом слове, полученном в предыдущем разделе, шестой бит изменил своё значение с нуля на единицу (на рисунке обозначено темным цветом). Тогда получим следующую матрицу синдромов.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |  |  |
| ***k0*** | ***k1*** | ***m0*** | ***k2*** | ***m1*** | ***m2*** | ***m3*** | ***k3*** | ***m4*** | ***m5*** | ***m6*** | ***m7*** | ***m8*** | ***m9*** | ***m10*** |  |  |
| 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 1 |  |  |
| 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | ***S0*** | 0 |
| 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | ***S1*** | 1 |
| 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | ***S2*** | 1 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | ***S3*** | 0 |

Заметим, что при однократной ошибке матрица синдромов всегда представляет собой двоичную запись (младший разряд в верхней стоке) номера позиции, в которой произошла ошибка. В приведённом примере матрица синдромов (0110) соответствует двоичному числу 0110 или десятичному 6, откуда следует, что ошибка произошла в шестом бите.

**Описание моделирующей программы**

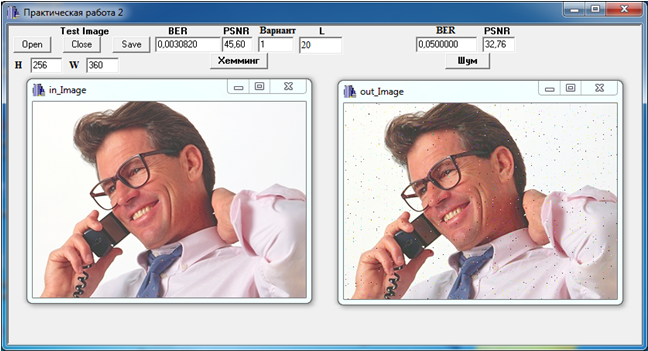
Для выполнения исследований корректирующей способности кодов Хемминга в качестве источника информации выбраны статические изображения. На рис. 2.4 показано интерфейсное окно моделирующей программы.

Рис. 2.4. Интерфейсное окно моделирующей программы

Программа выполняет следующие функции:

1. С помощью кнопки «Open» выполняется открытие папки, выбирается и загружается файл изображения в формате \*.bmp. При этом в окошках «Н» и «W» появляются значения высоты и ширины в пикселях.

2. Кнопка «Close» используется для очистки и закрытия окон с изображениями.

3. Кнопка «Save» позволяет сохранить файл изображения «out\_Image».

3. При нажатии кнопки «Шум» выполняются следующие операции:

* преобразование формата изображения из *RGB* в компоненты яркости *Y* и цветности *U*, *V*;
* преобразование байтовых массивов *Y*, *U* и *V* в битовые массивы;
* разбиение битового массива на интервалы с длиной *L*.
* наложение в каждом интервале *L* битового массива случайным образом единичных бит ошибок.
* Вычисление коэффициента битовых ошибок:   
  BER = (*NY* + *NU* + *NV*)/(24×H×W), где *NY* , *NU* и *NV* количество ошибок в битовых массивах *Y*, *U* и *V*.
* преобразование битовых массивов в байтовые массивы  и вывод зашумленного изображения в окно «out\_Image».
* вычисление отношения сигнал/шум (*PSNR*) по следующей формуле

(дБ).

где

,

(*YUV*) ˗ значения яркости и цветности пикселей исходного изображений; () ˗ значения пикселей восстановленного изображения.

4. При нажатии кнопки «Хемминг» выполняются следующие операции:

* преобразование формата изображения из *RGB* в компоненты яркости *Y* и цветности *U*, *V*;
* преобразование байтовых массивов *Y*, *U* и *V* в битовые массивы;
* кодирование битовых массивов кодом Хемминга по выбранному варианту (1, 2, 3….8);
* разбиение битового массива на интервалы с длиной *L*;
* наложение в каждом интервале *L* битового массива случайным образом единичных бит ошибок;
* декодирование кода Хемминга и восстановление битовых массивов *Y*, *U* и *V*;
* вычисление коэффициента битовых ошибок BER;
* преобразование битовых массивов в байтовые массивы , вычисление значения *PSNR* и вывод зашумленного изображения в окно «out\_Image».

**Порядок выполнения работы**

Перед началом выполнения работы каждый студент получает свой вариант задания. Варианты задания соответствует следующим кодам Хемминга.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| № | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| ***код*** | (7,4) | (15,11) | (31,26) | (63,57) | (67,60) | (127,120) | (130,122) | (255,247) |

1. Запустить приложение «PRAKTIKA\_2» и открыть файл \*.bmp тестового изображения. При отсутствии файлов формата \*.bmp можно использовать файлы другого формата после их сохранения в \*.bmp с помощью любого фоторедактора.
2. В окне «Вариант» установить заданный номер варианта.
3. Установить минимальное значение интервала ***Lmin*** равным 10, нажать кнопку «**Шум**» и записать значения ***BER*** и ***PSNR***. Для этого же интервала нажать кнопку «**Хемминг**» и записать значения ***BER***\_**1** и ***PSNR***\_**1**.
4. Изменяя интервал ***L***, найти максимальное значение ***Lmax*** при котором шумы, при использовании кода Хемминга, будут практически незаметны глазом.
5. Для найденного значения ***Lmax*** оценить: ***BER\_*1**, ***PSNR\_*1** и соответствующие значения ***BER*** и ***PSNR*** без применения помехоустойчивого кодирования. Рассчитать выигрыш от помехоустойчивого кодирования ***q*** = ***BER/ BER\_*1**.
6. Изменяя интервал ***L*** от ***Lmin***до ***Lmax***, построить таблицу для ***десяти***значений ***L***: ***BER*** = *F*(***L***), ***PSNR*** = *F*(***L***) без помехоустойчивого кодирования и ***BER***\_**1** = *F*(***L***), ***PSNR***\_***1*** = *F*(***L***) с применением кода Хемминга. Рассчитать и добавить в таблицу значения ***q*** = *F*(***L***).
7. Повторить пункты 1-6 для 3-х тестовых изображений, получить усредненные данные, построить таблицу зависимостей: ***BER*** = *F*(***L***), ***PSNR*** = *F*(***L***), ***BER***\_**1** = *F*(***L***), ***PSNR***\_**1** = *F*(***L***) и ***q*** = *F*(***L***).
8. Для полученных усредненных данных построить график ***BER***\_**1** = *F*(***L***).

**Требования к оформлению отчёта**

Отчёт должен содержать:

* титульный лист;
* задание на практическую работу;
* результаты выполнения программы: привести примеры *двух* зашумленных изображений с минимальным ***Lmin*** и максимальным значением ***Lmax*** без кодирования и с применением кода Хемминга.
* таблицы с зависимостями ***BER*** = *F*(***L***), ***PSNR*** = *F*(***L***), ***BER***\_**1** = *F*(***L***), ***PSNR***\_**1** = *F*(***L***) и ***q*** = *F*(***L***) для трех тестовых изображений. Пример оформления таблицы для одного изображения приведен ниже.

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ***L*** | 10 | .. | .. | .. | .. | .. | .. | .. | .. | ***Lmax*** | **Image** |
| ***BER*** |  |  |  |  |  |  |  |  |  |  | 1 |
| ***PSNR*** |  |  |  |  |  |  |  |  |  |  |
| ***BER***\_**1** |  |  |  |  |  |  |  |  |  |  |
| ***PSNR***\_**1** |  |  |  |  |  |  |  |  |  |  |
| ***q*** |  |  |  |  |  |  |  |  |  |  |

* таблица c усредненными данными по трем изображениям: ***BER*** = *F*(***L***), ***PSNR*** = *F*(***L***), ***BER***\_**1** = *F*(***L***), ***PSNR***\_**1** = *F*(***L***) и ***q*** = *F*(***L***);
* график усредненной зависимости ***BER***\_**1** = *F*(***L***).

**Контрольные вопросы**

1. Рассчитайте скорость кода и избыточность для заданного варианта.
2. Поясните, что собой представляет линейный блочный систематический код.
3. Способы оценки коэффициента битовых ошибок (BER).
4. Основные характеристики блочных кодов.
5. Порождающая матрица.
6. Поясните работу кодера Хемминга (***7***,***4***) по структурной схеме.
7. Проверочная матрица.
8. Принципы синдромного декодирования.
9. Алгоритм кодирования.
10. Алгоритм декодирования.
11. Как зависит вероятность ошибки от ее кратности?
12. Как зависит вероятность ошибки декодирования от вероятности ошибок канала связи?
13. С какой кратностью ошибок справляются коды Хемминга?