Государственный Комитет связи, информатизации и телекоммуникационных технологий Республики Узбекистан

Ташкентский университет информационных технологий

Кафедра «Информационная безопасность»

Курсовая работа

По предмету: «Теория информации и кодирование»

Тема: «Код Хемминга. Избыточность сообщений»

Группа: 205-13

Выполнил: Хусанов Х.

Проверил: Гуломов Ш.

Ташкент 2014

Содержание

Введение

Как положительную тенденцию, отвечающую требованиям сегодняшнего дня, следует отметить ускоренный рост услуг связи и информатизации, которые за год возросли на 41,6 процента. Это обеспечено в первую очередь за счет увеличения количества абонентов, пользующихся услугами мобильной связи и сети Интернет, чему способствовало принятие мер в отчетном году по снижению тарифов для населения на услуги по предоставлению доступа в Интернет на 22 процента. Сегодня около 8 миллионов человек являются активными пользователями сети Интернет.

В истекшем году большое внимание уделялось проведению активной инвестиционной политики, направленной на ускорение модернизации, технического и технологического перевооружения действующих и создание новых, современных, высокотехнологичных производств [1].

Хэмминг часто работал в выходные дни, и все больше и больше раздражался, потому что часто был должен перезагружать свою программу из-за ненадежности перфокарт. На протяжении нескольких лет он проводил много времени над построением эффективных алгоритмов исправления ошибок. В 1950 году он опубликовал способ, который известен как код Хэмминга.

В середине 1940-х годов Ричард Хэмминг работал в знаменитых лабораториях фирмы Белл (Bell Labs) на счётной машине Bell Model V. Это была электромеханическая машина, использующая релейные блоки, скорость которых была очень низка: один оборот за несколько секунд. Данные вводились в машину с помощью перфокарт, поэтому в процессе чтения часто происходили ошибки. В рабочие дни использовались специальные коды, чтобы обнаруживать и исправлять найденные ошибки, при этом оператор узнавал об ошибке по свечению лампочек, исправлял и снова запускал машину. В выходные дни, когда не было операторов, при возникновении ошибки машина автоматически выходила из программы и запускала другую.[3]

Код Хэмминга используется в некоторых прикладных программах в области хранения данных, особенно в RAID 2; кроме того, метод Хэмминга давно применяется в памяти типа ECC и позволяет «на лету» исправлять однократные и обнаруживать двукратные ошибки.

Постановка задачи: Построение алгоритма и разработка программы вычисление кода Хемминга.

1.1 Самокорректирующиеся коды

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

$$2^k \geq k + m$$

или

$$k \geq \log_2(k + m + 1),$$

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

Диапазон m	**k_{min}**
1	2
2-4	3
5-11	4
12-26	5
27-57	6

В настоящее время наибольший интерес представляют двоичные блочные корректирующие коды. При использовании таких кодов информация передаётся в виде блоков одинаковой длины и каждый блок кодируется и декодируется независимо друг от друга. Почти во всех блочных кодах символы можно разделить на информационные и проверочные. Таким образом, все комбинации кодов разделяются на разрешенные (для которых соотношение информационных и проверочных символов возможно) и запрещенные.

Основными характеристиками самокорректирующихся кодов являются:

1. Число разрешенных и запрещенных комбинаций. Если n - число символов в блоке, r - число проверочных символов в блоке, k - число информационных символов, то 2^n - число возможных кодовых комбинаций 2^k, - число разрешенных кодовых комбинаций, - 2^n -2^k число запрещенных комбинаций.
2. Избыточность кода. Величину r/n называют избыточностью корректирующего кода.
3. Минимальное кодовое расстояние. Минимальным кодовым расстоянием d 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 $d \geq 2g + 1$
5. Корректирующие возможности кодов.[8]

Граница Плоткина даёт верхнюю границу кодового расстояния

$$d \leqslant \frac{n*2^{k-1}}{2^k-1}$$

или при

$$n \geq 2 * d - 1$$

.

Граница Хэмминга устанавливает максимально возможное число разрешенных кодовых C_n^i комбинаций

$$2^k \leq 2^n / \sum_{i=0}^{\frac{d-1}{2}} C_n^i$$

где - число сочетаний из n элементов по i элементам. Отсюда можно получить выражение для оценки числа проверочных символов:

$$r \geq log_2(\sum_{i=0}^{\frac{d-1}{2}} C_n^i)$$

Для значений $d/n = \leq 0.3$ разница между границей Хэмминга и границей Плоткина невелика.

Граница Варшамова-Гильберта для больших *n* определяет нижнюю границу числа проверочных символов

$$r \geq log_2(\sum_{i=0}^{d-2} C_{n-1}^i)$$

.

Все вышеперечисленные оценки дают представление о верхней границе *d* при фиксированных n и k или оценку снизу числа проверочных символов.

1.2 Код Хэмминга

Код Хэмминга— это алгоритм самоконтролирующегося и самокорректирующегося кода, который позволяет закодировать какое-либо информационное сообщение определённым образом и после передачи(например, по сети) определить появилась ли какая-то ошибка в этом сообщении(к при-меру из-за помех) и, при возможности, восстановить это сообщение. В данном примере описан са-мый простой алгоритм Хэмминга, который может исправлять лишь одну ошибку(существуют более совершенные модификации данного алгоритма, которые позволяют обнаруживать(и если возможно исправлять) большее количество ошибок).

Код Хэмминга состоит из двух частей. Первая часть кодирует исходное сообщение, вставляя в него в определённых местах контрольные биты(вычисленные особым образом). Вторая часть получает входящее сообщение и заново вычисляет контрольные биты(по тому же алгоритму, что и первая часть). Если все вновь вычисленные контрольные биты совпадают с полученными, то сообщение получено без ошибок. В противном случае, выводится сообщение об ошибке и при возможности ошибка исправляется.

Построение кодов Хэмминга основано на принципе проверки на четность числа единичных символов: к последовательности добавляется такой элемент, чтобы число единичных символов в получившейся последовательности было четным.

$$r_1 = i_1 \oplus i_2 \oplus ... \oplus i_k.$$

знак $\oplus$ здесь означает сложение по модулю 2

$$S = i_1 \oplus i_2 \oplus ... \oplus i_n \oplus r_1.$$

$S = 0$ - ошибки нет, $s = 1$ однократная ошибка. Такой код называется $(k+1, k)$ или $(n, n-1)$.

Первое число - количество элементов последовательности, второе - количество информационных символов. Для каждого числа проверочных символов $r = 3, 4, 5..$ существует классический код Хэмминга с маркировкой $(n, k) = (2^r - 1, 2^r - 1 - r)$ т.е. - $(7,4), (15,11), (31,26)$. При

иных значениях k получается так называемый усеченный код, например международный телеграфный код МТК-2, у которого $k = 5$. Для него необходим код Хэмминга $(9, 5)$, который является усеченным от классического $(15, 11)$. Для Примера рассмотрим классический код Хэмминга $(7, 4)$. [4]

Сгруппируем проверочные символы следующим образом:

$$r_1 = i_1 \oplus i_2 \oplus i_3$$
$$r_2 = i_2 \oplus i_3 \oplus i_4$$
$$r_3 = i_1 \oplus i_2 \oplus i_4$$

знак $\oplus$ здесь означает сложение по модулю 2.

Получение кодового слова выглядит следующим образом:

$$\begin{pmatrix} i_1 & i_2 & i_3 & i_4 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 \end{pmatrix} = \begin{pmatrix} i_1 & i_2 & i_3 & i_4 & r_1 & r_2 & r_3 \end{pmatrix}$$

На вход декодера поступает кодовое слово

$$V = (i'_1, i'_2, i'_3, i'_4, r'_1, r'_2, r'_3)$$

где штрихом помечены символы, которые могут исказиться в результате помехи. В декодере в режиме исправления ошибок строится последовательность синдромов:

$$S_1 = r_1 \oplus i_1 \oplus i_2 \oplus i_3$$
$$S_2 = r_2 \oplus i_2 \oplus i_3 \oplus i_4$$
$$S_3 = r_3 \oplus i_1 \oplus i_2 \oplus i_4$$

$S = (S_1, S_2, S_3)$ называется синдромом последовательности.

Получение синдрома выглядит следующим образом:

$$(i_1 \quad i_2 \quad i_3 \quad i_4 \quad r_1 \quad r_2 \quad r_3)\begin{pmatrix}1 & 0 & 1\\ 1 & 1 & 1\\ 1 & 1 & 0\\ 0 & 1 & 1\\ 1 & 0 & 0\\ 0 & 1 & 0\\ 0 & 0 & 1\end{pmatrix} = (S_1 \quad S_2 \quad S_3)$$

Кодовые слова (7,4) кода Хэмминга

i_1	i_2	i_3	i_4	r_1	r_2	r_3
0	0	0	0	0	0	0
0	0	0	1	0	1	1
0	0	1	0	1	1	0
0	0	1	1	1	0	1
0	1	0	0	1	1	1
0	1	0	1	1	0	0
0	1	1	0	0	0	1
0	1	1	1	0	1	0
1	0	0	0	1	0	1
1	0	0	1	1	1	0
1	0	1	0	0	1	1
1	0	1	1	0	0	0
1	1	0	0	0	1	0
1	1	0	1	0	0	1
1	1	1	0	1	0	0
1	1	1	1	1	1	1

Синдром (0,0,0) указывает на то, что в последовательности нет искажений. Каждому ненулевому синдрому соответствует определенная конфигурация ошибок, которая исправляется на этапе декодирования. Для кода (7,4) в таблице указаны ненулевые синдромы и соответствующие им конфигурации ошибок (для вида: $i_1\, i_2\, i_3\, i_4\, r_3\, r_2\, r_1$).

Синдром	001	010	011	100	101	110	111
Конфигурация ошибок	0000001	0000010	0001000	0000100	1000000	0010000	0100000
Ошибка в символе	r_3	r_2	i_4	r_1	i_1	i_3	i_2

2.2 Избыточность сообщений

Избыточность - термин из теории информации, означающий превышение количества информации, используемой для передачи или хранения сообщения, над его информационной энтропией. Для уменьшения избыточности применяется сжатие данных без потерь, в то же время контрольная сумма применяется для внесения дополнительной избыточности в поток, что позволяет производить исправление ошибок при передаче информации по каналам, вносящим искажения (спутниковая трансляция, беспроводная передача и т. д.).[13]

Избыточность кода — это количество проверочной информации в сообщении.

Целесообразность устранения избыточности сообщения методами эффективного кодирования с последующим перекодированием помехоустойчивым кодом обусловлена тем, что избыточность источника сообщения в большинстве случаев не согласована со статистическими закономерностями помехи в канале связи и поэтому не может быть полностью использована для повышения достоверности принимаемого сообщения, тогда как обычно можно подобрать подходящий помехоустойчивый код. Кроме того, избыточность источника сообщений часто является следствием весьма сложных вероятностных зависимостей и позволяет обнаруживать и исправлять ошибки только после декодирования всего сообщения, пользуясь сложнейшими алгоритмами и интуицией.

Количественное определение

Информационное содержание одного сообщения в потоке, в наиболее общем случае, определяется как:

$$r = \mathbb{E}H(M_t|M_{t-1}, M_{t-2}, M_{t-3}, \dots)$$

Обозначим как R логарифм числа символов в алфавите сообщений:

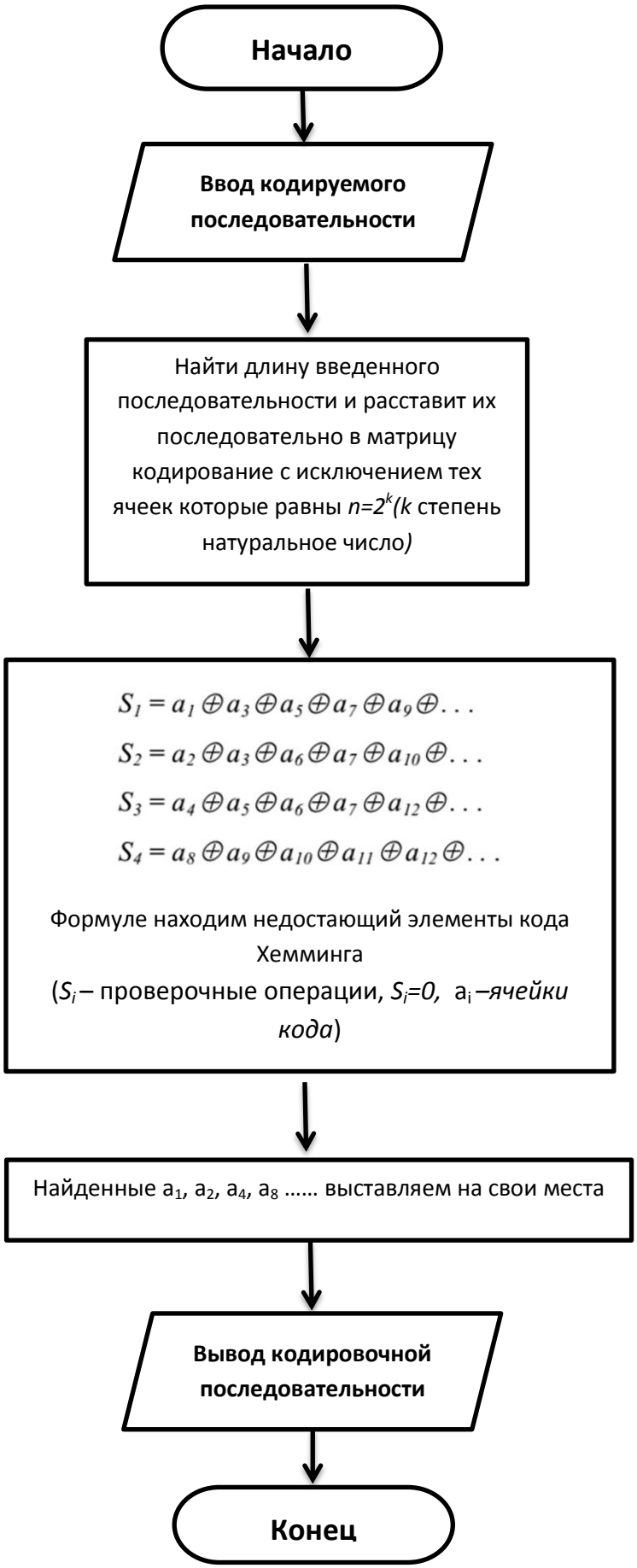
$$R = \log|M|$$

Абсолютная избыточность может быть определена как разность этих двух величин:

$$D = R - r$$

Соотношение $\boldsymbol{\frac{D}{R}}$ называется **относительной избыточностью** и дает математическую оценку максимальной степени сжатия, на которую может быть уменьшен размер файла.

2.2 Алгоритм построение кода Хэмминга



Пример построение кода Хэмминга

Предположим, что нужно сгенерировать код Хэмминга для некоторого информационного кодового слова. В качестве примера возьмём 15-битовое кодовое слово $\mathbf{x}_1...\mathbf{x}_{15}$, 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**	**12**	**13**	**14**	**15**
$\mathbf{x_1}$	$\mathbf{x_2}$	$\mathbf{x_3}$	$\mathbf{x_4}$	$\mathbf{x_5}$	$\mathbf{x_6}$	$\mathbf{x_7}$	$\mathbf{x_8}$	$\mathbf{x_9}$	$\mathbf{x_{10}}$	$\mathbf{x_{11}}$	$\mathbf{x_{12}}$	$\mathbf{x_{13}}$	$\mathbf{x_{14}}$	$\mathbf{x_{15}}$
1	0	0	1	0	0	1	0	1	1	1	0	0	0	1

Вставим в информационное слово контрольные биты $\mathbf{r}_0...\mathbf{r}_4$ таким образом, чтобы номера их позиций представляли собой целые степени двойки: 1, 2, 4, 8, 16... Получим 20-разрядное слово с 15 информационными и 5 контрольными битами. Первоначально контрольные биты устанавливаем равными нулю. На рисунке контрольные биты выделены розовым цветом.

1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**
$\mathbf{r_0}$	$\mathbf{r_1}$	$\mathbf{x_1}$	$\mathbf{r_2}$	$\mathbf{x_2}$	$\mathbf{x_3}$	$\mathbf{x_4}$	$\mathbf{r_3}$	$\mathbf{x_5}$	$\mathbf{x_6}$	$\mathbf{x_7}$	$\mathbf{x_8}$	$\mathbf{x_9}$	$\mathbf{x_{10}}$	$\mathbf{x_{11}}$	$\mathbf{r_4}$	$\mathbf{x_{12}}$	$\mathbf{x_{13}}$	$\mathbf{x_{14}}$	$\mathbf{x_{15}}$
0	0	1	0	0	0	1	0	0	0	1	0	1	1	1	0	0	0	0	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 5 строк (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**	**16**	**17**	**18**	**19**	**20**	
$\mathbf{r_0}$	$\mathbf{r_1}$	$\mathbf{x_1}$	$\mathbf{r_2}$	$\mathbf{x_2}$	$\mathbf{x_3}$	$\mathbf{x_4}$	$\mathbf{r_3}$	$\mathbf{x_5}$	$\mathbf{x_6}$	$\mathbf{x_7}$	$\mathbf{x_8}$	$\mathbf{x_9}$	$\mathbf{x_{10}}$	$\mathbf{x_{11}}$	$\mathbf{r_4}$	$\mathbf{x_{12}}$	$\mathbf{x_{13}}$	$\mathbf{x_{14}}$	$\mathbf{x_{15}}$	
0	0	1	0	0	0	1	0	0	0	1	0	1	1	1	0	0	0	0	1	
1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	$\mathbf{r_0}$
0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	$\mathbf{r_1}$
0	0	0	1	1	1	1	0	0	0	0	1	1	1	1	0	0	0	0	1	$\mathbf{r_2}$
0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	0	0	0	0	0	$\mathbf{r_3}$
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	$\mathbf{r_4}$

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, r_0) 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.[6]

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.

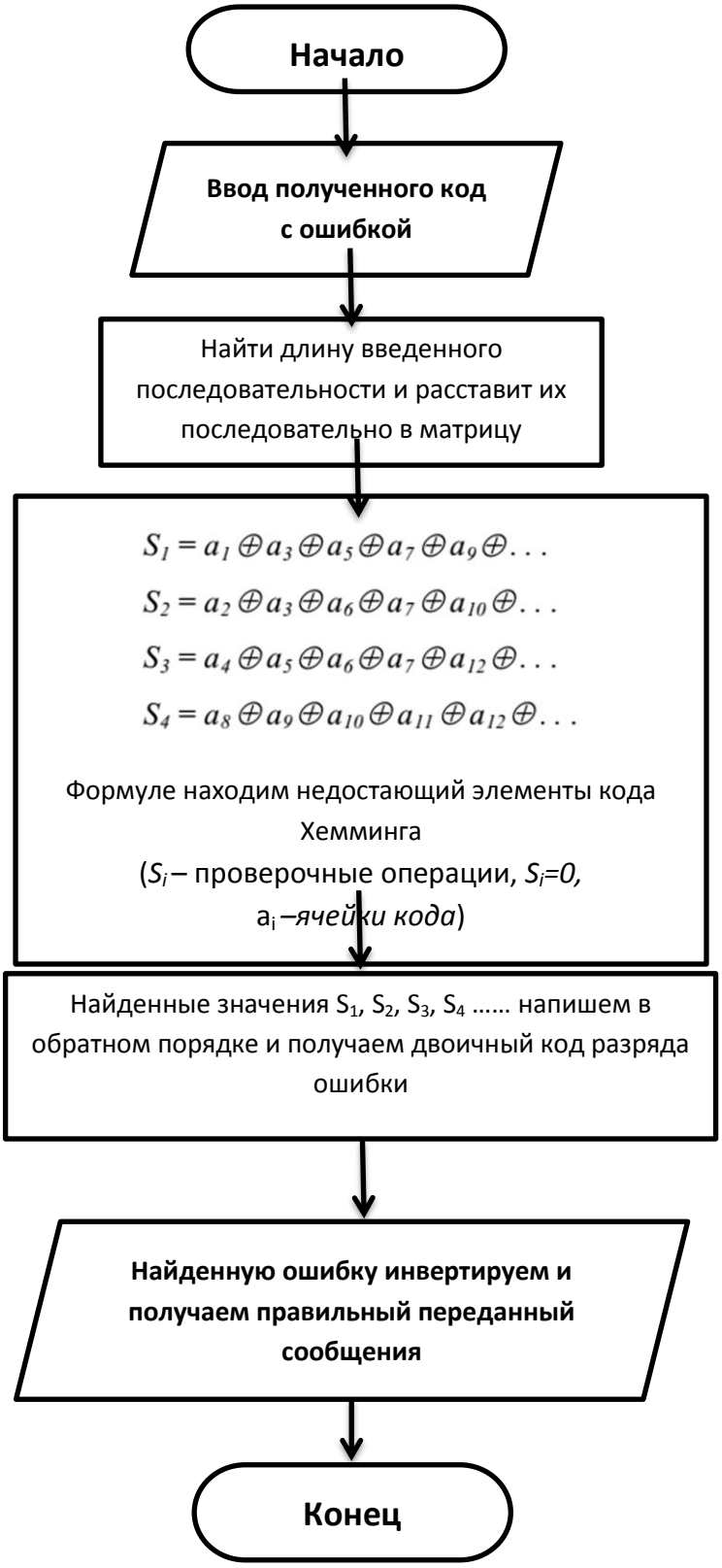
Например, для строки r_0:

$r_0 = (1 \cdot 0 + 0 \cdot 0 + 1 \cdot 1 + 0 \cdot 0 + 1 \cdot 0 + 0 \cdot 0 + 1 \cdot 1 + 0 \cdot 0 + 1 \cdot 0 + 0 \cdot 0 + 1 \cdot 1 + 0 \cdot 0 + 1 \cdot 1 + 0 \cdot 1 + 1 \cdot 1 + 0 \cdot 0 + 1 \cdot 0 + 0 \cdot 0 + 1 \cdot 0 + 0 \cdot 1) \bmod 2 = 5 \bmod 2 = 1.$

Полученные контрольные биты вставляем в кодовое слово вместо стоявших там ранее нулей. Кодирование по Хэммингу завершено. Полученное кодовое слово — 11110010001011110001.

1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**	
$\mathbf{r_0}$	$\mathbf{r_1}$	$\mathbf{x_1}$	$\mathbf{r_2}$	$\mathbf{x_2}$	$\mathbf{x_3}$	$\mathbf{x_4}$	$\mathbf{r_3}$	$\mathbf{x_5}$	$\mathbf{x_6}$	$\mathbf{x_7}$	$\mathbf{x_8}$	$\mathbf{x_9}$	$\mathbf{x_{10}}$	$\mathbf{x_{11}}$	$\mathbf{r_4}$	$\mathbf{x_{12}}$	$\mathbf{x_{13}}$	$\mathbf{x_{14}}$	$\mathbf{x_{15}}$	
1	1	1	1	0	0	1	0	0	0	1	0	1	1	1	1	0	0	0	1	
1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	$\mathbf{r_0}$ 1
0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	$\mathbf{r_1}$ 1
0	0	0	1	1	1	1	0	0	0	0	1	1	1	1	0	0	0	0	1	$\mathbf{r_2}$ 1
0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	0	0	0	0	0	$\mathbf{r_3}$ 0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	$\mathbf{r_4}$ 1

2.3 Алгоритм определение ошибки кода Хэмминга



Пример определение ошибки кода Хэмминга

Алгоритм определение ошибки 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.[5]

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 красным цветом). Тогда получим следующую матрицу синдромов.

1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**	
$\mathbf{r_0}$	$\mathbf{r_1}$	$\mathbf{x_1}$	$\mathbf{r_2}$	$\mathbf{x_2}$	$\mathbf{x_3}$	$\mathbf{x_4}$	$\mathbf{r_3}$	$\mathbf{x_5}$	$\mathbf{x_6}$	$\mathbf{x_7}$	$\mathbf{x_8}$	$\mathbf{x_9}$	$\mathbf{x_{10}}$	$\mathbf{x_{11}}$	$\mathbf{r_4}$	$\mathbf{x_{12}}$	$\mathbf{x_{13}}$	$\mathbf{x_{14}}$	$\mathbf{x_{15}}$	
1	1	1	1	0	1	1	0	0	0	1	0	1	1	1	1	0	0	0	1	
1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	$\mathbf{s_0}$ 0
0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	$\mathbf{s_1}$ 1
0	0	0	1	1	1	1	0	0	0	0	1	1	1	1	0	0	0	0	1	$\mathbf{s_2}$ 1
0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	0	0	0	0	0	$\mathbf{s_3}$ 0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	$\mathbf{s_4}$ 0

Заметим, что при однократной ошибке матрица синдромов всегда представляет собой двоичную запись (младший разряд в верхней строке) номера позиции, в которой произошла ошибка. В приведённом примере матрица синдромов (01100) соответствует двоичному числу 00110 или десятичному 6, откуда следует, что ошибка произошла в шестом бите.

Заключение

В данной курсовой работе было рассмотрено самокорректирующиеся коды и код Хэмминга, а также алгоритм построение и определение ошибки кода Хэмминга. В работе изучено основные понятие теории самокорректирующих кодов и этапы построение и определение ошибки кода Хэмминга. Практической части курсовой работы было создано алгоритм построение и определение кода Хэмминга, а также разработан программа и алгоритм программы.

Целесообразность устранения избыточности сообщения методами эффективного кодирования с последующим перекодированием помехоустойчивым кодом обусловлена тем, что избыточность источника сообщения в большинстве случаев не согласована со статистическими закономерностями помехи в канале связи и поэтому не может быть полностью использована для повышения достоверности принимаемого сообщения, тогда как обычно можно подобрать подходящий помехоустойчивый код. Кроме того, избыточность источника сообщений часто является следствием весьма сложных вероятностных зависимостей и позволяет обнаруживать и исправлять ошибки только после декодирования всего сообщения, пользуясь сложнейшими алгоритмами и интуицией.

Коды Хэмминга — наиболее известные и, вероятно, первые из самоконтролирующихся и самокорректирующихся кодов. Построены они применительно к двоичной системе счисления.

Литература

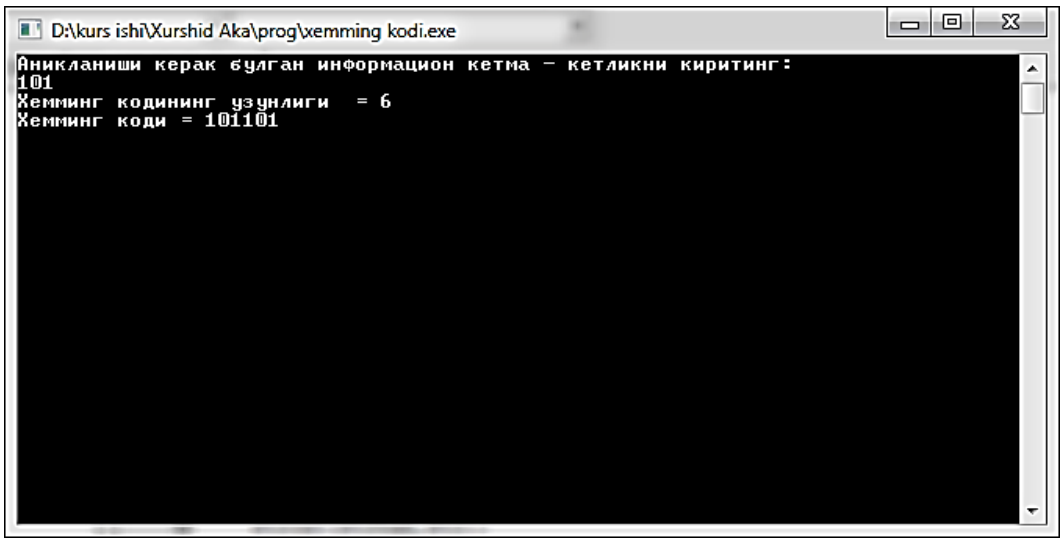
1. Доклад Президента Республики Узбекистан Ислама Каримова на заседании Кабинета Министров, посвященном основным итогам 2011 года и приоритетным направлениям социально-экономического развития Узбекистана на 2012 год.
2. Питерсон У., Уэлдон Э. Коды, исправляющие ошибки: Пер. с англ. М.: Мир, 1976, 600 с.
3. Пенин П.Е.,Филиппов Л.Н. Радиотехнические системы передачи информации. М.: Радио и Связь, 1984, 256 с.
4. Блейхут Р. Теория и практика кодов, контролирующих ошибки. Пер. с англ. М.: Мир, 1986, 576 с.
5. Г.И. Никитин. Первичные коды: Метод. указ., ЛИАП, Л., 1984, 28 с.
6. Г.И. Никитин. Эффективные коды: Метод. указ., ЛИАП, Л., 1987, 28 с.
7. А.К. Журавлев, Г.И. Никитин. Радиотехнические системы передачи информации: Учеб.пособие/ЛИАП, Л., 1984, 86 с.
8. Питерсон У., Уэлдон Э. Коды, исправляющие ошибки: Пер. с англ. М.: Мир, 1976, 600 с.
9. Кларк Д., Кейн Д. Кодирование с исправлением ошибок в системах цифровой связи: Пер. с англ. М.: Радио и Связь,1987, 300 с.
10. Пенин П.Е.,Филиппов Л.Н. Радиотехнические системы передачи информации. М.: Радио и Связь, 1984, 256 с.
11. Блейхут Р. Теория и практика кодов, контролирующих ошибки. Пер. с англ М.: Мир, 1986, 576 с.
12. http://ru.wikipedia.org
13. http://ru.wikipedia.org/w/index.php?title=Избыточность_информации&o l di d=62027173
14. http://www.ngpedia.ru

Приложение

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#include<math.h>
#include<iostream>

using namespace std;

char Data[10000];
char ch;


void read ()

{
    ch=getchar();
}

int main (void)
{
    int i=1;
    int S=0;
    int z=0;
    int p=0;
    char *str=new char[100];
    char *strham=new char[100];
    char s[2]="0";
    setlocale (LC_ALL, "Russian");
```

```
        cout<<"Аникланиши керак булган информацион
кетма – кетликни киритинг:\n";
        read();

        while (ch!='\n')
        {
            while
(fmod(log((float)i)/log(2.0),1.0)<0.0000001)
            {
                ++i;
            }

            Data[i]=ch;
            ++i;
            read();
        }
                cout<<"Хемминг кодининг узунлиги =
"<<i-1<<endl;
        for (int q=1; q<i; ++q)
        {
            if (Data[q]=='1') S=S^q;
        }
        itoa(S,str,2);

        for (int q=1; q<i; ++q)
        {
            if (Data[q]!='1' && Data[q]!='0') ++z;
        }
        int len=strlen(str);
```

```
    for (int q=0; q<(z-len); ++q)
    {
        strcat(strham,s);
    }
    strcat(strham,str);

    for (int q=1; q<i; ++q)
    {
        if (Data[q]!='1' && Data[q]!='0')
        {
            Data[q]=strham[strlen(strham)-1-p];
            ++p;
        }
    }
    cout<<"Хемминг коди = ";
    for (int q=1; q<i; ++q)
    {
        cout<<Data[q];
    }

    fgetc(stdin);

    delete []str;
    delete []strham;
}
```


D:\kurs ishi\Xurshid Aka\prog\xemming kodi.exe
Аникланиши керак булган информацион кетма – кетликни киритинг:
101
Хемминг кодининг узунлиги = 6
Хемминг коди = 101101