

Код Хэмминга. Пример работы алгоритма

Понятие о корректирующих кодах

- Обрабатываемая информация представляется различными комбинациями из двух символов 0 и 1; поэтому любой процесс кодирования состоит из преобразования чисел и слов в соответствующие последовательности символов 1 и 0.
- *Код* – это совокупность всех комбинаций из определенного количества символов (кодowego алфавита), которые избраны для представления информации. Каждая такая комбинация называется *кодовой комбинацией*.
- Общее число кодовых комбинаций в данном коде может быть равно или меньше числа всех возможных комбинаций из данного количества символов.

коды равномерные и

неравномерные

- *Равномерные* – коды, в которых все комбинации имеют одинаковое количество знаков.
- *Неравномерные* – коды, в которых количество знаков может быть различным. Примером такого кода может служить телеграфный код Морзе.
- При помощи n двоичных знаков, очевидно, можно получить 2^n кодовых комбинаций. В зависимости от того, все возможные 2^n кодовые комбинации задействованы для представления информации или нет, коды подразделяются на **простые** и

Простые коды

- *Простые* – коды, в которых используются все возможные 2^n комбинации, полученные при помощи n двоичных знаков.
- В таком коде всякая ошибка, состоящая в изменении 0 на 1 или 1 на 0, превращает одну информационную комбинацию в другую. Для обнаружения и исправления ошибки в таком коде необходима дополнительная информация.
- *Пример.* Пусть $n=3$. Тогда количество возможных кодовых комбинаций $2^n = 8$. Простой код для $n=3$ будет иметь следующий вид:

000

001

010

011

100

101

110

111

Корректирующие коды

- *Корректирующие* – коды, в которых лишь некоторая часть всех возможных 2^n комбинаций, полученных при помощи n двоичных знаков, используется для представления информации.
- В таком коде все остальные кодовые комбинации являются запрещенными, и их появление свидетельствует о наличии ошибки. Любая одиночная ошибка в таком коде превращает информационную комбинацию в запрещенную.
- *Пример.* Пусть $n=3$, но из всех возможных кодовых комбинаций, представленных в предыдущем примере, только четыре изображают числа от 0 до 3, а остальные считаются запрещенными. Такой корректирующий код будет иметь следующий вид:

000

011

101

110

Корректирующие коды

- Корректирующие коды можно разделить на *систематические* и *несистематические*.
- *Систематические* – такие n -значные коды, которые содержат постоянное количество m информационных и $k = n - m$ избыточных знаков, причем эти знаки занимают одни и те же позиции во всех кодовых комбинациях.
- *Несистематические* – такие коды, в которых знаки закодированного числа или слова разделить на информационные и контрольные невозможно.

Корректирующие коды

- Основными характеристиками корректирующих кодов являются их *избыточность* и *корректирующая способность*.

Избыточность кода определяется по формуле $k = n - m$,

где n – общее число знаков в коде;

m – число информационных знаков, необходимых для изображения

N чисел или слов в простом коде, определяемое по формуле

$$m = \log_2 N;$$

k – число контрольных знаков.

Например, для кода из *примера*: $m = \log_2 4 = 2$; $k = 1$.

Часто для характеристики кода применяют понятие *относительная избыточность*, которая

Корректирующие коды

- **Корректирующая способность** кода количественно может быть определена вероятностью обнаружения или исправления ошибок различных типов. Разработка кодов, имеющих максимальную корректирующую способность при заданной избыточности, а также кодов, обеспечивающих заданную корректирующую способность при минимальной избыточности, – одна из важнейших задач теории кодирования.
- Корректирующая способность кода связана с понятием кодового расстояния. Прежде чем сформулировать определение кодового расстояния, введем понятие о весе кодовой комбинации.
- **Вес, $W(A)$** , кодовой комбинации A определяется количеством содержащихся в ней двоичных единиц.
- **Пример:** для $A = 111001$, $W(A) = \sum a_i = 4$.

Корректирующие коды

- **Кодовое расстояние** между двумя кодовыми комбинациями определяется числом позиций, в которых их элементы не совпадают.
- Это означает, что кодовое расстояние между комбинациями A и B равно весу некоторой третьей комбинации C , полученной поразрядным сложением двух этих комбинаций в соответствии со следующей формулой:

$$W(C) = W(A \oplus B) = \sum (a_i \oplus b_i).$$

- **Пример.** Пусть есть кодовая комбинация $A = 111001$ и кодовая комбинация $B = 100101$, тогда

$$\begin{array}{r} 111001 \\ \oplus 100101 \\ \hline \end{array}$$

$$C = 011100 \quad W(C) = 3, \text{ кодовое расстояние между } A \text{ и } B$$

Корректирующие коды

- Минимальное кодовое расстояние α кода – это минимальное расстояние между двумя любыми комбинациями в этом коде.
- Если, например, в коде есть хотя бы одна пара комбинаций, которые отличаются друг от друга только в одной позиции, то минимальное расстояние этого кода $\alpha = 1$.
Так, для простого кода из первого примера $\alpha = 1$, а для корректирующего кода из второго примера $\alpha = 2$.

Рассмотрим некоторые корректирующие коды.

Код с проверкой на четность

- Простейший корректирующий код – код с проверкой на четность, который образуется добавлением к группе информационных разрядов одного избыточного, значение которого выбирается таким образом, чтобы сумма единиц в кодовой комбинации, т. е. вес кодовой комбинации, была всегда четна.
- *Пример.* Рассмотрим код с проверкой на четность, образованный добавлением контрольного разряда к простому коду из *примера*.

	Информационные разряды	Контрольный разряд
0	000	0
1	001	1
2	010	1
3	011	0
4	100	1
5	101	0
6	110	0
7	111	1

Код с проверкой на четность

- Таким образом, если в простом коде число 4 имеет изображение 100, то в коде с проверкой на четность оно будет изображаться комбинацией 1001.
- Минимальное кодовое расстояние кода с проверкой на четность $\alpha = 2$. Такой код обнаруживает все одиночные ошибки и групповые ошибки нечетной кратности, так как четность количества единиц в этом случае будет также нарушаться.
- Следует отметить, что при кодировании целесообразно число единиц в кодовой комбинации делать нечетным и проводить контроль на нечетность, в этом случае любая комбинация, в том числе и изображающая нуль, будет иметь хотя бы одну единицу, что дает возможность отличить полное отсутствие информации от передачи нуля.

Код Хэмминга

- Код Хэмминга представляет собой систематический код, имеющий большую относительную избыточность, нежели код с проверкой на четность и предназначен либо для исправления одиночных ошибок (при $\alpha = 3$), либо для исправления одиночных и обнаружения без исправления двойных ошибок (при $\alpha = 4$).
- N – значный код Хэмминга, имеет m информационных разрядов и k – контрольных.
- Число контрольных разрядов должно удовлетворять соотношению $k \geq \log_2 (n + 1)$,

Откуда

$$m \leq n - \log_2 (n + 1)$$

Код Хэмминга

- Код Хэмминга строится таким образом, что к имеющимся информационным разрядам кодовой комбинации добавляется вычисленное по вышеприведенной формуле количество контрольных разрядов, которые формируются путем подсчета четности суммы единиц для определенных групп информационных разрядов.
- При приеме такой кодовой комбинации из полученных информационных и контрольных разрядов путем аналогичных подсчетов четности составляют корректирующее число, которое равно нулю, при отсутствии ошибки, либо указывает номер ошибочного разряда.

- Рассмотрим подробнее процесс кодирования для кода с минимальным кодовым расстоянием $\alpha = 3$.
- Пусть первый контрольный разряд имеет нечетный порядковый номер. Установим его при кодировании таким образом, чтобы сумма единиц всех разрядов с нечетными порядковыми номерами была равна 0.

Такая операция может быть записана в виде соотношения

$$E_1 = a_1 \oplus a_3 \oplus a_5 \oplus a_7 \dots = 0,$$

- где a_1, a_3, a_5, a_7 – двоичные символы, размещенные в разрядах с номерами 1, 3, 5, 7, ...

Появление единицы во втором разряде (справа) корректирующего числа означает ошибку в тех разрядах кодовой комбинации, порядковые номера которых (2, 3, 6, 7, ...) в двоичном изображении имеют

- Поэтому вторая операция кодирования, позволяющая найти второй контрольный разряд, имеет вид

$$E_2 = a_2 \oplus a_3 \oplus a_6 \oplus a_7 \dots = 0.$$

- Рассуждая аналогичным образом,

$$E_3 = a_4 \oplus a_5 \oplus a_6 \oplus a_7 \dots = 0;$$

$$E_4 = a_8 \oplus a_9 \oplus a_{10} \oplus a_{11} \dots = 0.$$

- После приема кодовой комбинации, совместно со сформированными контрольными разрядами, выполняются те же операции подсчета, что были описаны выше, а полученное число $E_k E_{k-1} \dots E_2 E_1$ считается корректирующим, причем при $E_k E_{k-1} \dots E_2 E_1 = 0$ ошибки отсутствуют, а при наличии ошибок неравными нулю оказываются те суммы E_i , в образовании которых участвовал ошибочный разряд.
- Корректирующее число при этом будет равно порядковому номеру этого разряда.

- Выбор места для контрольных разрядов в каждой из кодовых комбинаций определяется таким образом, чтобы контрольные разряды участвовали только в одной операции подсчета четности.
- Такими позициями являются целые степени двойки: 1, 2, 4, 8, 16,
- *Пример*. Составим шестизначный код Хэмминга $n = 6$,
 $k \geq \log_2 7$, $k = 3$, $m = n - k = 3$.

Цифра	Простой код	Код Хэмминга					
		6	5	4	3	2	1
0	000	0	0	0	0	0	0
1	001	0	0	0	1	1	1
2	010	0	1	1	0	0	1
3	011	0	1	1	1	1	0
4	100	1	0	1	0	1	0
5	101	1	0	1	1	0	1
6	110	1	1	0	0	1	1
7	111	1	1	0	1	0	0

- Принят код: 111100 исправлено 110100 – ошибка по корректирующему числу в разряде 4;
- 111010 исправлено 101010 – ошибка по корректирующему числу в разряде 5;
- 100000 исправлено 000000 – ошибка по корректирующему числу в разряде 6

Цифра	Простой код	Код Хэмминга					
		6	5	4	3	2	1
0	000	0	0	0	0	0	0
1	001	0	0	0	1	1	1
2	010	0	1	1	0	0	1
3	011	0	1	1	1	1	0
4	100	1	0	1	0	1	0
5	101	1	0	1	1	0	1
6	110	1	1	0	0	1	1
7	111	1	1	0	1	0	0

К существующим k контрольным разрядам может еще добавляться $(k+1)$ -й разряд, обеспечивающий дополнительный контроль по четности всей кодовой комбинации.

При проверке информации после ее приема возможны три случая:

- отсутствие ошибок – корректирующее число равно 0, общая четность суммы единиц кодовой комбинации правильна;
- одиночная ошибка – контроль общей четности кодовой комбинации обнаруживает ошибку, корректирующее число указывает номер искаженного разряда (если корректирующее число равно нулю, то ошибка произошла в разряде общей четности);
- двойная ошибка – корректирующее число не равно нулю, контроль общей четности кодовой комбинации не обнаруживает ошибки

- Код Хэмминга - это алгоритм, 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 примеру из-за помех) и, при возможности, восстановить это сообщение.
- Таким образом, код Хэмминга - самоконтролирующийся и самокорректирующийся код. Построен он применительно к двоичной системе счисления.

- Рассмотрим самый простой алгоритм Хемминга, который может исправлять лишь одну ошибку.
- Существуют модификации данного алгоритма, позволяющие 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. Иначе делается вывод о наличии ошибки

Логика алгоритма

- Допустим, есть сообщение из двух символов: «ha», которое необходимо передать без ошибок. Чтобы сообщение закодировать при помощи Кода Хэмминга, сначала необходимо представить его в бинарном

Символ	ASCII код	Бинарное представление
h	68	01000100
a	61	00111101

- Надо определиться с длиной информационного слова, то есть длиной строки из нулей и единиц, которая будет кодироваться. Допустим, длина слова будет равна 16.
- Таким образом, любое исходное сообщение необходимо разделить на блоки по 16 бит, которые будут потом кодироваться отдельно друг от друга.
- Т.к. один символ занимает 8 бит, то в кодируемое слово помещается два ASCII символа. В

h	a
01000100	00111101

- Необходимо вставить **контрольные биты**. Они вставляются в строго определённых местах — это **позиции с номерами, равными степеням двойки**.
- В нашем случае (при длине информационного слова в 16 бит) это будут позиции 1, 2, 4, 8, 16. Соответственно, получилось 5 контрольных бит (выделены красным цветом):

h	a
000010000100	001011101

- Таким образом, длина всего сообщения увеличилась на 5 бит. До вычисления самих контрольных бит им присваивается значение «0»

Вычисление контрольных бит.

- Значение каждого контрольного бита зависит от значений информационных бит, но не от всех, а только от тех, которые этот контрольных бит контролирует.
- Для того, чтобы понять, за какие биты отвечает каждый контрольный бит, необходимо понять простую закономерность:
контрольный бит с номером N контролирует все последующие N бит

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	
0	0	0	0	1	0	0	0	0	1	0	0	0	0	1	0	1	1	1	0	1	
X		X		X		X		X		X		X		X		X		X		X	1
	X	X			X	X			X	X			X	X			X	X			2
			X	X	X	X					X	X	X	X					X	X	4
							X	X	X	X	X	X	X	X							8
															X	X	X	X	X	X	16

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	
0	0	0	0	1	0	0	0	0	1	0	0	0	0	1	0	1	1	1	0	1	
X		X		X		X		X		X		X		X		X		X		X	1
	X	X			X	X			X	X			X	X			X	X			2
			X	X	X	X					X	X	X	X					X	X	4
							X	X	X	X	X	X	X	X							8
															X	X	X	X	X	X	16

- Здесь знаком «X» обозначены те биты, которые контролирует контрольный бит, номер которого справа.
- То есть, к примеру, бит номер 12 контролируется битами с номерами 4 и 8.
- Чтобы узнать какими битами контролируется бит с номером N надо разложить N по степеням двойки.

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	
0	0	0	0	1	0	0	0	0	1	0	0	0	0	1	0	1	1	1	0	1	
X		X		X		X		X		X		X		X		X		X		X	1
	X	X			X	X			X	X			X	X			X	X			2
			X	X	X	X					X	X	X	X					X	X	4
							X	X	X	X	X	X	X	X							8
															X	X	X	X	X	X	16

- Как же вычислить значение каждого контрольного бита?
- Делается это просто: берут каждый контролируемый бит и смотрят, сколько среди контролируемых им битов единиц, получают некоторое целое число и, если оно чётное, то ставят ноль, в противном случае ставят единицу.

h	a
100110000100	001011101

Декодирование и исправление ошибок

- Теперь, допустим, закодированное сообщение пришло с ошибкой. Пусть 11-ый бит передан неправильно:

h	a
100110000110	001011101

- Алгоритм декодирования заключается в том, что необходимо заново вычислить все контрольные биты (так же как при кодировании) и сравнить их с контрольными битами, которые были получены.

Посчитав контрольные биты, получаем:

h	a
010110010110	001011101

- Контрольные биты под номерами: 1, 2, 8 не совпадают с такими же контрольными битами, которые были получены. Теперь, сложив номера позиций неправильных контрольных бит ($1 + 2 + 8 = 11$), получаем позицию ошибочного бита.

Параметры кода Хэмминга

- Параметры кода указываются так: (7, 4). Это означает, что длина кодового слова равна 7 битам, а длина сообщения – 4 бита.
- В зависимости от количества информационных и проверочных разрядов в кодовых словах существуют коды Хэмминга (7,4), (9,5), (11, 7), (15, 11), (31, 26), (63, 57) и т. д.
- Общий вид формулы, по которой определяются виды кодов Хэмминга по соотношению числа информационных символов к проверочным: $(2^x - 1, 2^x - x - 1)$, где x – натуральное число.
- При декодировании есть вероятность, что исходное сообщение нельзя будет восстановить, в случае превышения числом ошибок корректирующей способности кода.
- Однако помехоустойчивость закодированной

- Требуется написать кодировщик, который будет получать на вход 11 бит данных, кодировать их и возвращать 15 бит выходной информации.

Т.е. параметр кода Хэмминга (15, 11).

Возвращает кодер 16 бит (кодированное слово):

- кодированное слово дополняется одним битом слева, чтобы длина была равна степени двойки. Сейчас этот бит никак не используется. Можно его использовать как бит чётности и получить так называемый дополненный код Хэмминга.

Постановка задачи. Кодер.

Кодирует 11 бит сообщения кодом Хэмминга (15, 11).

Входные данные – 11 бит, выходные – 16 бит.

Размещение проверочных и информационных бит в кодовом слове:

	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
in_data								L	K	I	H	G	F	E	D	C B A
code_word		L		K		I	H	G	F	E	P	D	C	B	P	A P P X

A,B,C,D,E,F,G,H,I,K,L – биты данных информационного слова;

P – проверочный бит;

X – бит, равный 0 (не используется).

Кодовое слово дополняется одним битом, чтобы длина была равна степени двойки.

```
Dim preDataIn As New BitArray(11)
```

```
For i As Integer = 0 To 10
```

```
    If (b.Count > i) Then preDataIn(i) = b(i)
```

```
    Else Exit For
```

```
End If
```

```
Next
```

```
Dim dataIn As New BitArray(preDataIn)
```

'процесс кодирования – сложение по модулю 2 бит информационного слова.

```
Dim codeWord As New BitArray(16)
```

'Младший разряд не используется.

```
codeWord(0) = False
```

'Вычисление первого проверочного символа:

```
codeWord(1) = dataIn(0) Xor dataIn(1) Xor dataIn(3) Xor dataIn(4) Xor  
dataIn(6) Xor dataIn(8) Xor dataIn(10)
```

'Вычисление второго проверочного символа:

```
codeWord(2) = dataIn(0) Xor dataIn(2) Xor dataIn(3) Xor dataIn(5) Xor  
dataIn(6) Xor dataIn(9) Xor dataIn(10)
```

'Вычисление третьего проверочного символа:

$\text{codeWord}(4) = \text{dataIn}(1) \text{ Xor } \text{dataIn}(2) \text{ Xor } \text{dataIn}(3) \text{ Xor } \text{dataIn}(7) \text{ Xor } \text{dataIn}(8) \text{ Xor } \text{dataIn}(9) \text{ Xor } \text{dataIn}(10)$

'Вычисление четвертого проверочного символа:

$\text{codeWord}(8) = \text{dataIn}(4) \text{ Xor } \text{dataIn}(5) \text{ Xor } \text{dataIn}(6) \text{ Xor } \text{dataIn}(7) \text{ Xor } \text{dataIn}(8) \text{ Xor } \text{dataIn}(9) \text{ Xor } \text{dataIn}(10)$

'Информационные символы:

$\text{codeWord}(3) = \text{dataIn}(0)$

$\text{codeWord}(5) = \text{dataIn}(1)$

$\text{codeWord}(6) = \text{dataIn}(2)$

$\text{codeWord}(7) = \text{dataIn}(3)$

$\text{codeWord}(9) = \text{dataIn}(4)$

$\text{codeWord}(10) = \text{dataIn}(5)$

$\text{codeWord}(11) = \text{dataIn}(6)$

$\text{codeWord}(12) = \text{dataIn}(7)$

$\text{codeWord}(13) = \text{dataIn}(8)$

$\text{codeWord}(14) = \text{dataIn}(9)$

$\text{codeWord}(15) = \text{dataIn}(10)$

Return codeWord

Постановка задачи. Декодер.

Декодер получает на вход 16 бит закодированных данных и возвращает 11 бит декодированных информационных данных, которые распределены по двум байтам.

Размещение проверочных и информационных бит в кодовом слове:

	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00		
code_word	L	K	I	H	G	F	E	P	D	C	B	P	A	P	P	X		
out_data								L	K	I	H	G	F	E	D	C	B	A

A,B,C,D,E,F,G,H,I,K,L – биты информационного слова;

P – проверочный бит;

X – бит, равный 0 (не используется)

Dim codeWord As New BitArray(b) '16 бит входных данных

' Процесс декодирования – это сложение по модулю 2 бит информационного слова, по весу полученных единиц в результате – получение позиции ошибки.

Dim syndrome As New BitArray(4)

'Вычисление первого проверочного символа из полученного кодового слова и далее сравнение его с полученным.

syndrome(0) = codeWord(3) Xor codeWord(5) Xor codeWord(7)
Xor codeWord(9) Xor codeWord(11) Xor codeWord(13) Xor
codeWord(15) Xor codeWord(1)

'Вычисление второго проверочного символа из полученного кодового слова и далее сравнение его с полученным.

syndrome(1) = codeWord(3) Xor codeWord(6) Xor codeWord(7)
Xor codeWord(10) Xor codeWord(11) Xor codeWord(14) Xor
codeWord(15) Xor codeWord(2)

'Вычисление третьего проверочного символа из полученного кодового слова и далее сравнение его с полученным.

syndrome(2) = codeWord(5) Xor codeWord(6) Xor codeWord(7)
Xor codeWord(12) Xor codeWord(13) Xor codeWord(14) Xor
codeWord(15) Xor codeWord(4)

'Вычисление четвёртого проверочного символа из полученного кодового слова и далее сравнение его с полученным.

```
syndrome(3) = codeWord(9) Xor codeWord(10) Xor codeWord(11) Xor  
codeWord(12) Xor codeWord(13) Xor codeWord(14) Xor  
codeWord(15) Xor codeWord(8)
```

'Вычисление по синдрому позиции ошибки. Это дешифратор.

'Если смотреть на синдром как на число - то это и есть номер позиции ошибки.

'Синдром равен 0 - ошибки нет.

'Поскольку на выход модуля передаются только биты данных - не все варианты перечислены, нет смысла исправлять проверочные биты.

```
Dim syn As Integer = (Convert.ToInt32(syndrome(3)) << 3) Or  
(Convert.ToInt32(syndrome(2)) << 2) Or  
(Convert.ToInt32(syndrome(1)) << 1) Or  
Convert.ToInt32(syndrome(0))
```

```
'          |15|14|13|12|11|10|09|08|07|06|05|04|03|02|01|00|  
'code_word | L | K | I | H | G | F | E | P | D | C | B | P | A | P | P | X |
```

'Синдромы ошибок в информационных битах, позиции в кодовом слове 3,5,6,7,9,10,11,12,13,14,15:

```
Dim correction As New BitArray(11)
Select Case syn
Case 3 'позиция 3
    correction = New BitArray({True, False, False, False, False, False, False, False, False, False, False})
Case 5 'позиция 5
    correction = New BitArray({False, True, False, False, False, False, False, False, False, False, False})
Case 6 'позиция 6
    correction = New BitArray({False, False, True, False, False, False, False, False, False, False, False})
Case 7 'позиция 7
    correction = New BitArray({False, False, False, True, False, False, False, False, False, False, False})
Case 9 'позиция 9
    correction = New BitArray({False, False, False, False, True, False, False, False, False, False, False})
Case 10 'позиция 10
    correction = New BitArray({False, False, False, False, False, True, False, False, False, False, False})
Case 11 'позиция 11
    correction = New BitArray({False, False, False, False, False, False, True, False, False, False, False})
Case 12 'позиция 12
    correction = New BitArray({False, False, False, False, False, False, False, True, False, False, False})
Case 13 'позиция 13
    correction = New BitArray({False, False, False, False, False, False, False, False, True, False, False})
Case 14 'позиция 14
    correction = New BitArray({False, False, False, False, False, False, False, False, False, True, False})
Case 15 'позиция 15
    correction = New BitArray({False, False, False, False, False, False, False, False, False, False, True})
Case Else
'Если синдром равен 0 или указывает на ошибку в проверочном символе "P" - коррекция
информационных символов не требуется.
'correction() = 0 при инициализации (correction = New BitArray(11)), поэтому ничего делать не
нужно.
End Select
```

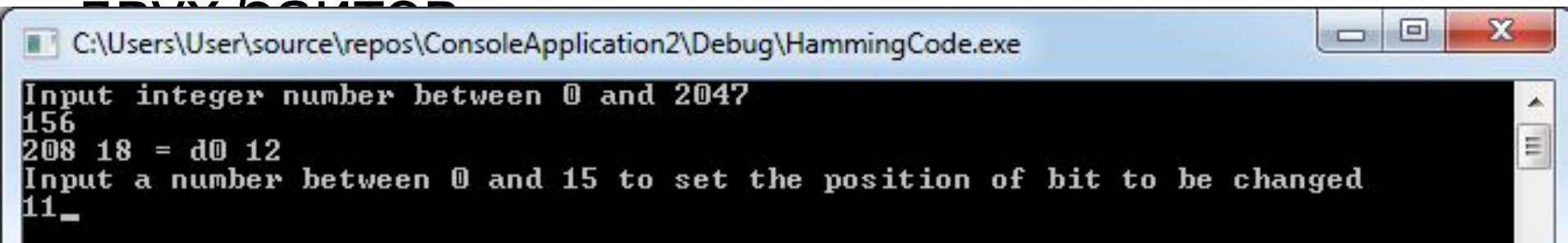
'Результат декодирования с учетом коррекции (11 бит выходных данных):

```
Dim outData As New BitArray(11)
outData(0) = codeWord(3) Xor correction(0)
outData(1) = codeWord(5) Xor correction(1)
outData(2) = codeWord(6) Xor correction(2)
outData(3) = codeWord(7) Xor correction(3)
outData(4) = codeWord(9) Xor correction(4)
outData(5) = codeWord(10) Xor correction(5)
outData(6) = codeWord(11) Xor correction(6)
outData(7) = codeWord(12) Xor correction(7)
outData(8) = codeWord(13) Xor correction(8)
outData(9) = codeWord(14) Xor correction(9)
outData(10) = codeWord(15) Xor correction(10)
Dim masks(31) As Integer
masks(0) = BitVector32.CreateMask()
For i As Integer = 1 To 31
    masks(i) = BitVector32.CreateMask(masks(i - 1))
Next
Dim v As New BitVector32
For i As Integer = 0 To 10
    v(masks(i)) = outData(i)
Next
Dim decoded As Integer = v.Data
Return decoded
```

Консольная программа, кодирующая и декодирующая код (15, 11)

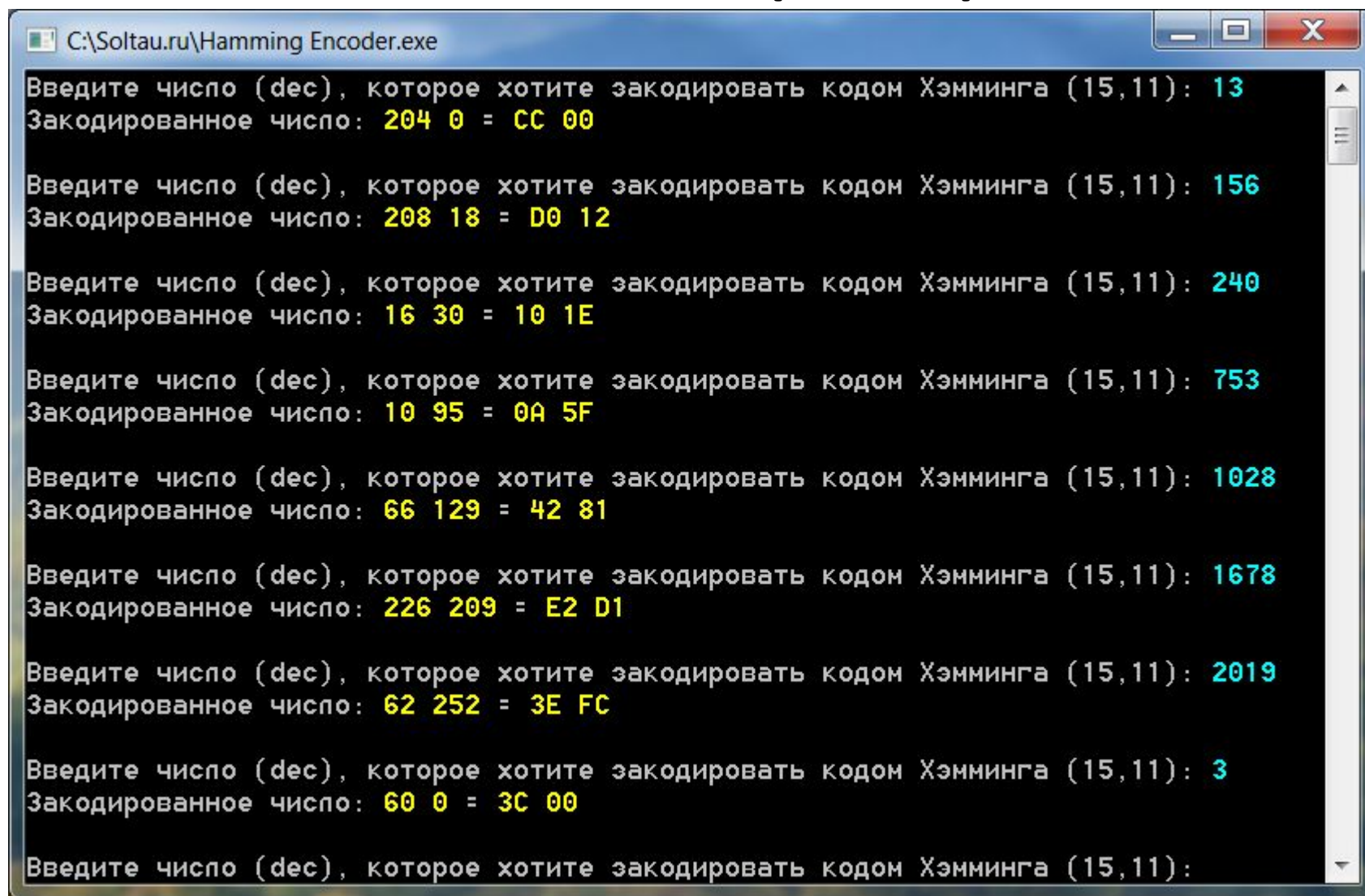
- Для проверки кодировщика и декодировщика кода Хэмминга (15, 11), используя вышеописанные функции, надо написать консольную программу, можно использовать любую среду разработки (Delphi, Visual Studio). Вводится 11-разрядное число (от 0 до 0x7FF или 2047), и на выходе получаем 16-разрядное число, представленное в виде

two bytes



```
C:\Users\User\source\repos\ConsoleApplication2\Debug\HammingCode.exe
Input integer number between 0 and 2047
156
208 18 = d0 12
Input a number between 0 and 15 to set the position of bit to be changed
11_
```

Консольная программа, кодирующая код Хэмминга (15, 11)



```
C:\Soltau.ru\Hamming Encoder.exe

Введите число (dec), которое хотите закодировать кодом Хэмминга (15,11): 13
Закодированное число: 204 0 = CC 00

Введите число (dec), которое хотите закодировать кодом Хэмминга (15,11): 156
Закодированное число: 208 18 = D0 12

Введите число (dec), которое хотите закодировать кодом Хэмминга (15,11): 240
Закодированное число: 16 30 = 10 1E

Введите число (dec), которое хотите закодировать кодом Хэмминга (15,11): 753
Закодированное число: 10 95 = 0A 5F

Введите число (dec), которое хотите закодировать кодом Хэмминга (15,11): 1028
Закодированное число: 66 129 = 42 81

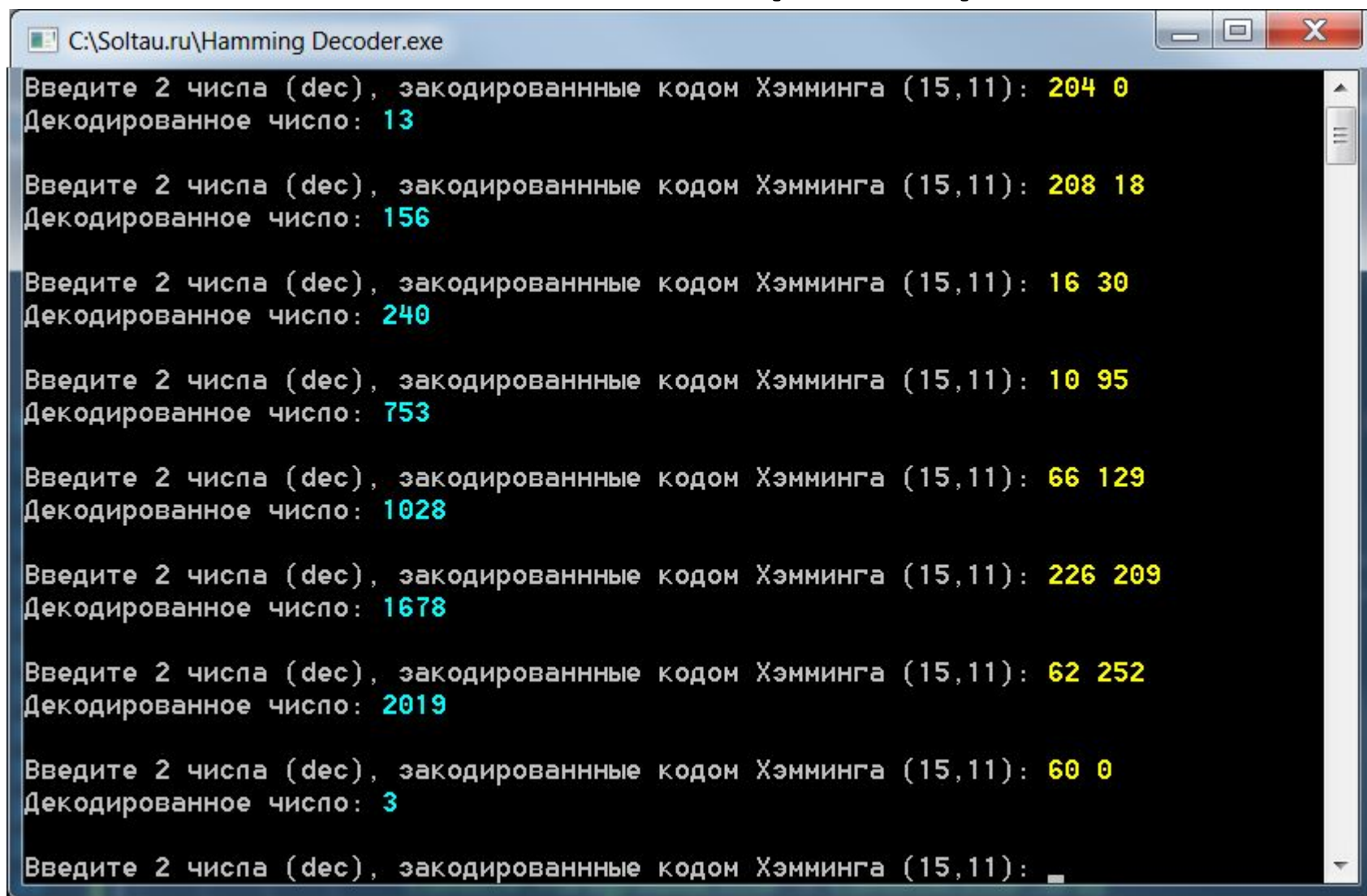
Введите число (dec), которое хотите закодировать кодом Хэмминга (15,11): 1678
Закодированное число: 226 209 = E2 D1

Введите число (dec), которое хотите закодировать кодом Хэмминга (15,11): 2019
Закодированное число: 62 252 = 3E FC

Введите число (dec), которое хотите закодировать кодом Хэмминга (15,11): 3
Закодированное число: 60 0 = 3C 00

Введите число (dec), которое хотите закодировать кодом Хэмминга (15,11):
```

Консольная программа, декодирующая код Хэмминга (15, 11)



```
C:\Soltau.ru\Hamming Decoder.exe

Введите 2 числа (dec), закодированные кодом Хэмминга (15,11): 204 0
Декодированное число: 13

Введите 2 числа (dec), закодированные кодом Хэмминга (15,11): 208 18
Декодированное число: 156

Введите 2 числа (dec), закодированные кодом Хэмминга (15,11): 16 30
Декодированное число: 240

Введите 2 числа (dec), закодированные кодом Хэмминга (15,11): 10 95
Декодированное число: 753

Введите 2 числа (dec), закодированные кодом Хэмминга (15,11): 66 129
Декодированное число: 1028

Введите 2 числа (dec), закодированные кодом Хэмминга (15,11): 226 209
Декодированное число: 1678

Введите 2 числа (dec), закодированные кодом Хэмминга (15,11): 62 252
Декодированное число: 2019

Введите 2 числа (dec), закодированные кодом Хэмминга (15,11): 60 0
Декодированное число: 3

Введите 2 числа (dec), закодированные кодом Хэмминга (15,11): _
```

При внесении битовой ошибки при декодировании, декодер восстанавливает исходное закодированное