Федеральное агентство по образованию
Государственное образовательное учреждение высшего
профессионального образования
«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»
Кафедра «Информационная безопасность систем и технологий»

ПОМЕХОУСТОЙЧИВОЕ КОДИРОВАНИЕ

Пояснительная записка к курсовой работе
по дисциплине «Передача дискретных сообщений»

ПГУ 3.090106.001 ПЗ

Руководитель КР,
д.т.н., профессор ______________ Б. В. Султанов

Исполнитель КР,
студент ______________ М. А. Захаров

Пенза 2009

«УТВЕРЖДАЮ»
Зав. кафедрой ИБСТ
______________С. Л. Зефиров
«____» ____________ 2009 г.

ЗАДАНИЕ

на курсовую работу

по теме: «Помехоустойчивое кодирование»
1 Дисциплина Передача дискретных сообщений
2 Вариант задания 7
3 Студент Захаров М. А. группа 06УИ1
4 Исходные данные на курсовую работу

- код Хемминга. Задача 1.1.7;
- код БЧХ. Задача 2.3.7;
- код Рида—Соломона. Задача 3.4.7.

5 Структура работы
5.1 Пояснительная записка (содержание работы):

- расчётно-пояснительная записка объёмом 15–20 страниц содержит 3 раздела (в соответствии с пунктами задания).

5.2 Графическая часть

- схема регистра кодирующего устройства;
- таблица состояний ячеек регистра (1 лист формата А4).

5.3 Экспериментальная часть

- не предусмотрена.

6 Календарный план выполнения работы
6.1 Сроки выполнения работ по разделам:

- раздел первый к 25.09.2009 г.

– раздел второй к 20.10.2009 г.

– раздел третий к 25.11.2009 г.

– оформление пояснительной записки к 06.12.2009 г.

Дата защиты работы 6 декабря 2009 г.

Руководитель работы Султанов Б. В.

Задание получил 7 сентября 2009 г.

Студент Захаров М. А.

Нормоконтролёр Султанов Б. В.

РЕФЕРАТ

Пояснительная записка 28 с., 0 рис., 0 табл., 1 источников, 2 прил.

ПОМЕХОУСТОЙЧИВОЕ КОДИРОВАНИЕ, КОД ХЕММИНГА, КОД БОУЗА—ЧОУДХУРИ—ХОКВИНГЕМА, КОД РИДА—СОЛОМОНА, OCTAVE

Объектом исследования являются помехоустойчивые коды.

Цель работы — решение задач по синтезу следующих помехоустойчивых кодов: код Хемминга, код Боуза—Чоудхури—Хоквингема и код Рида—Соломона.

В процессе выполнения курсовой работы были рассчитаны помехоустойчивые коды в соответствии с заданием. Для расчётов в полях $GF(q)$ была использована программа *Octave*.

В результате исследования были получены навыки по расчёту помехоустойчивых кодов с заданными характеристиками.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Текст введения.

1 КОД ХЕММИНГА

1.1 Условие задачи

Определить порождающий многочлен $g(x)$ кода Хемминга, скорость которого $R \geqslant r_0$, рассматривая его как код БЧХ, исправляющий одиночные ошибки. Сформировать разрешённую комбинацию систематического кода, соответствующую заданной информационной комбинации $a(x) = 10011000111$. Исправить ошибку в принимаемой кодовой комбинации $V'(x) = 111110001000010$.

1.2 Решение задачи

В данном случае код Хемминга имеет размерность (15,11), т. к. по условию скорость $R \geqslant r_0$. При $k = 11$ и $n = 15$, получаем, что $R = \frac{k}{n} = 0{,}73$, что превышает значение r_0, равное 0,7.

Так как код Хемминга предложено рассматривать как код БЧХ, то первым этапом при расчёте кода будет определение порождающего многочлена $g(x)$. Порождающий многочлен для кода БЧХ определяется из выражения:

$$g(x) = \text{НОК}\left[f_1(x), f_2(x) \ldots f_{2t_{\text{и}}}\right],$$

где НОК — наименьшее общее кратное;

$f_1(x), f_2(x), \ldots$ — минимальные многочлены корней $\alpha^1, \alpha^2 \ldots$ порождающего многочлена.

Корнем многочлена $g(x)$ называется число (элемент поля) при подстановке которого в выражение многочлена вместо x многочлен обращается в 0. Минимальный многочлен элемента β поля $GF(q^m)$ определяется из выражения:

$$f(x) = (x - \beta^{g^0})(x - \beta^{g^1}) \dots (x - \beta^{g^{l-1}}),$$

где l — наименьшее целое число, при котором:

$$\beta^{g^0} = \beta^{g^l}.$$

На практике для определения значения порождающего многочлена можно воспользоваться таблицей минимальных неприводимых многочленов в поле $GF(2^m)$, в которой приведены минимальные многочлены.

Для определения порождающего многочлена необходимо, во-первых, по заданной длине кода n определить из выражения $n = 2^m - 1$ значение параметра m, который является степенью сомножителя $g(x)$. Затем из выражения $j = 2t_{\text{и}} - 1$ определяем максимальный порядок минимального многочлена, входящих в число сомножителей $g(x)$. После этого, пользуясь таблицей минимальных многочленов, определяем выражение для $g(x)$, зависимости от найденных m и j. Для этого из колонки соответствующей параметру m выбираются многочлены с номерами от 1 до j, которые в результате перемножения дают выражение для $g(x)$. В нашем случае $n = 15$, а $t_{\text{и}} = 1$, следовательно пользуясь описанной выше методикой, получаем $m = 4$, $j = 1$, откуда получаем $g(x) = 010011$, или в виде многочлена $g(x) = x^4 + x + 1$.

Следующим этапом построения является построение производящей матрицы $G_{(n,k)}$. Для систематического кода матрица $G_{(n,k)}$ имеет вид:

$$G_{(n,k)} = \left[I_k R_{(k,r)} \right],$$

где I_k — единичная матрица размером $k \times k$.

Строки матрицы $R_{(k,r)}$ определяются из выражений:

$$r_i(x) = R_{g(x)}\big(a_i(x) \times x^r\big), \tag{1}$$

или

$$r_i(x) = R_{g(x)}\left(x^{n-i}\right),$$

где $a_i(x)$ — полином, соответствующий i-той строке матрицы I_k;

i — номер строки матрицы $R_{(k,r)}$;

$R_{g(x)}\big(a(x)\big)$ — остаток от деления $a(x)$ на $g(x)$.

Выполнив деление многочленов, согласно формуле (1), получаем:

$$R_{(15,11)} = \begin{bmatrix} 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix}.$$

В данном случае для кода (15,11) матрица I_k,будет иметь следующий вид:

$$I_{11} = \begin{bmatrix} 1&0&0&0&0&0&0&0&0&0&0\\ 0&1&0&0&0&0&0&0&0&0&0\\ 0&0&1&0&0&0&0&0&0&0&0\\ 0&0&0&1&0&0&0&0&0&0&0\\ 0&0&0&0&1&0&0&0&0&0&0\\ 0&0&0&0&0&1&0&0&0&0&0\\ 0&0&0&0&0&0&1&0&0&0&0\\ 0&0&0&0&0&0&0&1&0&0&0\\ 0&0&0&0&0&0&0&0&1&0&0\\ 0&0&0&0&0&0&0&0&0&1&0\\ 0&0&0&0&0&0&0&0&0&0&1 \end{bmatrix}.$$

Таким образом, производящая матрица будет иметь следующий вид:

$$G_{(15,11)} = \begin{bmatrix} 1&0&0&0&0&0&0&0&0&0&0&1&0&0&1\\ 0&1&0&0&0&0&0&0&0&0&0&1&1&0&1\\ 0&0&1&0&0&0&0&0&0&0&0&1&1&1&1\\ 0&0&0&1&0&0&0&0&0&0&0&1&1&1&0\\ 0&0&0&0&1&0&0&0&0&0&0&0&1&1&1\\ 0&0&0&0&0&1&0&0&0&0&0&1&0&1&0\\ 0&0&0&0&0&0&1&0&0&0&0&0&1&0&1\\ 0&0&0&0&0&0&0&1&0&0&0&1&0&1&1\\ 0&0&0&0&0&0&0&0&1&0&0&1&1&0&0\\ 0&0&0&0&0&0&0&0&0&1&0&0&1&1&0\\ 0&0&0&0&0&0&0&0&0&0&1&0&0&1&1 \end{bmatrix}.$$

Для получения кодовой комбинации необходимо вектор, соответствующий кодовой комбинации $a(x)$, умножить на матрицу $G_{(15,11)}$. Полученный в результате умножения вектор и будет являться разрешённой кодовой комбинацией. В соответствии с заданием $a(x) = 10011000111$. Выполним умножение:

$$V(x) = a(x) \times G_{(15,11)} = 10011000111 \times$$

$$\times \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 \end{bmatrix} =$$

$$= 100110001111001.$$

Проверочная матрица в систематическом виде строится на основе матрицы $G_{(n,k)}$, а именно:

$$H_{(n,k)} = \left[R^T_{(k,r)} I_r\right],$$

где I_r — единичная матрица;

$R^T_{(k,r)}$ — матрица $R_{(k,r)}$ из $G_{(n,k)}$ в транспонированном виде.

В соответствии с матрицей $G_{(15,4)}$ получаем:

$$H_{(15,11)} = \begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \end{bmatrix}.$$

Для определения синдрома необходимо умножить полученную кодовую комбинацию на $H^T_{(n,r)}$. Полученный в результате умножения вектор и будет являться синдромом, по которому можно судить о наличии и расположении ошибки, или её отсутствии. Принятая кодовая комбинация $V'(x) = 111110001000010$. Выполним умножение:

$$S(x) = V'(x) \times H^T_{(n,r)} = 111110001000010 \times \begin{bmatrix} 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = 1100.$$

В соответствии с этим синдромом определяем по матрице $H_{(15,4)}$, что ошибка произошла в 12 разряде, следовательно, исправленная комбинация будет 111010001000010.

1.3 Ответ

В результате решения были найдены образующий полином $g(x) = x^4+x+1$, разрешённая кодовая комбинация, соответствующая информационной комбинации a(x) — V(x) = 100110001111001, была определена ошибка в кодовой комбинации $V'(x) = 111110001000010$, после исправления была получена исправленная кодовая комбинация — 111010001000010.

2 КОД БОУЗА—ЧОУДХУРИ—ХОКВИНГЕМА

2.1 Условие задачи

Определить порождающий многочлен $g(x)$ примитивного кода БЧХ над $GF(2)$ длины $n = 2^m - 1$, исправляющего ошибки кратностью $t_и = 2$. Сформировать разрешённую комбинацию систематического кода, соответствующую заданной информационной комбинации:

$$a(x) = 10011100001111000000.$$

Построить регистр кодирующего устройства систематического циклического кода с порождающим многочленом $g(x)$, привести таблицу, иллюстрирующую состояние ячеек в процессе работы регистра при поступлении на его вход информационного блока $a(x)$. Определить, является ли разрешённой принимаемая кодовая комбинация:

$$V'(x) = 1101111011011011000110001010000.$$

2.2 Решение задачи

Первым этапом решения задачи по синтезу кода БЧХ является определение порождающего многочлена $g(x)$. Для этого необходимо воспользоваться методикой, описанной выше при решении задачи синтеза кода Хемминга. По условию задачи кратность исправляемых ошибок $t_и = 2$, следовательно в соответствии с формулой $j = 2t_и - 1$, получаем $j = 3$. Длина кода $n = 31$ и $m = 5$. Таким образом, полином $g(x)$ будет равен произведению полиномов записанных в первой, второй и третьей строках 4-го столбца таблицы минимальных многочленов, т. е.

$$g(x) = 45 \times 75 = (x^5 + x^2 + 1)(x^5 + x^4 + x^3 + x^2 + 1) = \\ = x^{10} + x^9 + x^8 + x^6 + x^5 + x^3 + 1.$$

В двоичном виде $g(x) = 11101101001$.

После нахождения порождающего многочлена, необходимо сформировать разрёшённую кодовую комбинацию систематического кода, соответствующую заданной информационной последовательности, по следующей формуле:

$$V(x) = a(x) \times x^r + r(x), \qquad (2)$$

где r — колличество проверочных разрядов;

$r(x)$ — остаток от деления $a(x) \times x^r$ на $g(x)$.

Таким образом, первые $n - r$ разрядов будут совпадать с информационной последовательностью, а последние r разрядов будут проверочными. По условию задачи $a(x) = 100111000011111000000$, а полиному x^{10} соответствует последовательность 10000000000, таким образом, получаем:

$$a(x) \times x^r = 1001110000111110000000000000000.$$

Остаток от деления $r(x)$ найден в программе *Octave*:

$$r(x) = 1111011111.$$

Таким образом, разрешённая кодовая комбинация по формуле (2):

$$V(x) = 100111000011111000000111011111.$$

Следующим этапом решения задачи является построение регистра кодирующего устройства и приведение таблицы переключений состояний ячеек регистра при поступлении на вход информационной последовательности $a(x)$. Схема регистра кодирующего устройства приведена в приложении А на рисунке А.1.

По условию задачи так же необходимо определить, является ли разрешённой кодовая комбинация:

$$V'(x) = 110111101101101100011000101000 0.$$

Для этого необходимо найти синдром $S(x)$, т.е. произвести деление многочлена $V'(x)$ на порождающий многочлен $g(x)$. Если остаток от деления будет равен нулю, то комбинация является разрешённой. В противном случае — неразрешённой. Операция производилась в программе *Octave*.

В результате выполнения деления, остаток равен 0, следовательно, принятая комбинация является разрешённой.

2.3 Ответ

В процессе решения задачи был определён порождающий многочлен:

$$g(x) = x^{10} + x^9 + x^8 + x^6 + x^5 + x^3 + 1;$$

сформирована разрешённая кодовая комбинация, соответствующая информационной последовательности $a(x)$:

$$V(x) = 100111000011111000000111011111;$$

построен регистр кодирующего устройства и построена таблица состояний ячеек регистра при поступлении на его вход информационной последовательности $a(x)$. Так же было определено, что принятая кодовая комбинация $V'(x)$ является разрешённой.

ЗАКЛЮЧЕНИЕ

Пример ссылки на ресурс [1].

Приложение А

(обязательное)

Регистр кодирующего устройства

Приложение Б

(обязательное)

Вычисления в среде GNU Octave

Нахождение остатка от деления для полинома в 1-м разделе:

```
[maxim@home-dekstop ~]$ octave
GNU Octave, version 3.2.3
Copyright (C) 2009 John W. Eaton and others.
This is free software; see the source code for copying
conditions. There is ABSOLUTELY NO WARRANTY; not even for
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
For details, type `warranty'.

Octave was configured for "i386-redhat-linux-gnu".

Additional information about Octave is available at http://
   www.octave.org.

Please contribute if you find this software useful.
For more information, visit http://www.octave.org/help-wanted
   .html

Report bugs to <bug@octave.org> (but first, please read
http://www.octave.org/bugs.html to learn how to write a
   helpful report).

For information about changes from previous versions, type `
   news'.

warning: mark_as_command is obsolete and will be removed from
    a future version of Octave
octave:1> p1 = gf([1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
   0], 2);
octave:2> p2 = gf([1, 0, 0, 1, 1], 2);
octave:3> [chastn, remd] = deconv(p1, p2)
chastn =
GF(2^2) array. Primitive Polynomial = D^2+D+1 (decimal 7)
```

```
Array elements =

   1   0   0   1   1   0   1   0   1   1   1

remd =
GF(2^2) array. Primitive Polynomial = D^2+D+1 (decimal 7)

Array elements =

   0   0   0   0   0   0   0   0   0   0   0   1   0   0   1

octave:4> p1 = gf([1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
    2);
octave:5> [chastn, remd] = deconv(p1, p2)
chastn =
GF(2^2) array. Primitive Polynomial = D^2+D+1 (decimal 7)

Array elements =

   1   0   0   1   1   0   1   0   1   1

remd =
GF(2^2) array. Primitive Polynomial = D^2+D+1 (decimal 7)

Array elements =

   0   0   0   0   0   0   0   0   0   0   1   1   0   1

octave:6> p1 = gf([1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0], 2)
   ;
octave:7> [chastn, remd] = deconv(p1, p2)
chastn =
GF(2^2) array. Primitive Polynomial = D^2+D+1 (decimal 7)

Array elements =

   1   0   0   1   1   0   1   0   1
```

```
remd =
GF(2^2) array. Primitive Polynomial = D^2+D+1 (decimal 7)

Array elements =

   0   0   0   0   0   0   0   0   0   1   1   1   1

octave:8> p1 = gf([1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0], 2);
octave:9> [chastn, remd] = deconv(p1, p2)
chastn =
GF(2^2) array. Primitive Polynomial = D^2+D+1 (decimal 7)

Array elements =

   1   0   0   1   1   0   1   0

remd =
GF(2^2) array. Primitive Polynomial = D^2+D+1 (decimal 7)

Array elements =

   0   0   0   0   0   0   0   0   1   1   1   0

octave:10> p1 = gf([1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0], 2);
octave:11> [chastn, remd] = deconv(p1, p2)
chastn =
GF(2^2) array. Primitive Polynomial = D^2+D+1 (decimal 7)

Array elements =

   1   0   0   1   1   0   1

remd =
GF(2^2) array. Primitive Polynomial = D^2+D+1 (decimal 7)

Array elements =

   0   0   0   0   0   0   0   0   1   1   1
```

```
octave:12> p1 = gf([1, 0, 0, 0, 0, 0, 0, 0, 0, 0], 2);
octave:13> [chastn, remd] = deconv(p1, p2)
chastn =
GF(2^2) array. Primitive Polynomial = D^2+D+1 (decimal 7)

Array elements =

   1   0   0   1   1   0

remd =
GF(2^2) array. Primitive Polynomial = D^2+D+1 (decimal 7)

Array elements =

   0   0   0   0   0   0   1   0   1   0

octave:14> p1 = gf([1, 0, 0, 0, 0, 0, 0, 0, 0], 2);
octave:15> [chastn, remd] = deconv(p1, p2)
chastn =
GF(2^2) array. Primitive Polynomial = D^2+D+1 (decimal 7)

Array elements =

   1   0   0   1   1

remd =
GF(2^2) array. Primitive Polynomial = D^2+D+1 (decimal 7)

Array elements =

   0   0   0   0   0   0   1   0   1

octave:16> p1 = gf([1, 0, 0, 0, 0, 0, 0, 0], 2);
octave:17> [chastn, remd] = deconv(p1, p2)
chastn =
GF(2^2) array. Primitive Polynomial = D^2+D+1 (decimal 7)

Array elements =
```

```
   1   0   0   1

remd =
GF(2^2) array. Primitive Polynomial = D^2+D+1 (decimal 7)

Array elements =

   0   0   0   0   1   0   1   1

octave:18> p1 = gf([1, 0, 0, 0, 0, 0, 0], 2);
octave:19> [chastn, remd] = deconv(p1, p2)
chastn =
GF(2^2) array. Primitive Polynomial = D^2+D+1 (decimal 7)

Array elements =

   1   0   0

remd =
GF(2^2) array. Primitive Polynomial = D^2+D+1 (decimal 7)

Array elements =

   0   0   0   1   1   0   0

octave:20> p1 = gf([1, 0, 0, 0, 0, 0], 2);
octave:21> [chastn, remd] = deconv(p1, p2)
chastn =
GF(2^2) array. Primitive Polynomial = D^2+D+1 (decimal 7)

Array elements =

   1   0

remd =
GF(2^2) array. Primitive Polynomial = D^2+D+1 (decimal 7)

Array elements =
```

```
   0    0    0    1    1    0

octave:22> p1 = gf([1, 0, 0, 0, 0], 2);
octave:23> [chastn, remd] = deconv(p1, p2)
chastn =
GF(2^2) array. Primitive Polynomial = D^2+D+1 (decimal 7)

Array elements =

   1

remd =
GF(2^2) array. Primitive Polynomial = D^2+D+1 (decimal 7)

Array elements =

   0    0    0    1    1

octave:24> quit

[maxim@home-dekstop ~]$
```

Нахождение разрешённой комбинации систематического кода и остатков от деления полиномов для 2 раздела:

```
[maxim@home-dekstop ~]$ octave

octave:1> a = gf([1, 0, 0, 1, 1, 1, 0, 0, 0, 0, 1, 1, 1, 1,
   1, 0, 0, 0, 0, 0, 0], 2);
octave:2> x_r =  gf([1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0], 2);
octave:3> proizv = conv(a, x_r)
proizv =
GF(2^2) array. Primitive Polynomial = D^2+D+1 (decimal 7)

Array elements =

 Columns 1 through 22:

 1 0 0 1 1 1 0 0 0 0 1 1 1 1 1 0 0 0 0 0 0 0
```

```
 Columns 23 through 31:

 0 0 0 0 0 0 0 0 0

octave:4> g = gf([1, 1, 1, 0, 1, 1, 0, 1, 0, 0, 1], 2);
octave:5> [chastn, remd] = deconv(proizv, bg)
chastn =
GF(2^2) array. Primitive Polynomial = D^2+D+1 (decimal 7)

Array elements =

 1 1 0 0 0 1 0 0 1 0 0 0 1 1 1 0 0 0 1 1 1

remd =
GF(2^2) array. Primitive Polynomial = D^2+D+1 (decimal 7)

Array elements =

 Columns 1 through 22:

 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1

 Columns 23 through 31:

 1 1 1 0 1 1 1 1 1

octave:6> sum = proizv + remd
sum =
GF(2^2) array. Primitive Polynomial = D^2+D+1 (decimal 7)

Array elements =

 Columns 1 through 22:

 1 0 0 1 1 1 0 0 0 0 1 1 1 1 1 0 0 0 0 0 0 1

 Columns 23 through 31:

 1 1 1 0 1 1 1 1 1
```

```
octave:7> v = gf([1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 0,
   1, 1, 0,
0, 0, 1, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0],2);
octave:8> [chastn, remd] = deconv (v, g)
chastn =
GF(2^2) array. Primitive Polynomial = D^2+D+1 (decimal 7)

Array elements =

 1 0 1 0 1 1 0 1 1 0 1 0 0 1 1 0 1 0 0 0 0

remd =
GF(2^2) array. Primitive Polynomial = D^2+D+1 (decimal 7)

Array elements =

 Columns 1 through 22:

 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

 Columns 23 through 31:

 0 0 0 0 0 0 0 0 0

octave:9> quit
```

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Гонтмахер Е. Ш. Национальные проекты: первые итоги реализации // SPERO. — 2008. — №8. — С. 119–134. — Систем. требования: Adobe Reader. URL: http://spero.socpol.ru/docs/N8_2008-119-134.pdf