**МИНОБРНАУКИ РОССИИ**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ  
ВЫСШЕГО ОБРАЗОВАНИЯ  
**«БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ  
ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ им. В.Г.ШУХОВА»  
(БГТУ им. В.Г.Шухова)**

Кафедра программного обеспечения вычислительной техники и автоматизированных систем

Лабораторная работа №6  
дисциплина: Теория информации  
тема: «Помехоустойчивое кодирование. Циклические коды»

Выполнил: ст. группы ВТ-22

Воскобойников Илья Сергеевич

Белгород 2020

**Задания к работе**

1. Разложить многочлен x7+1 на множители. Получить порождающий и проверочный многочлены.
2. Взять произвольное ненулевое двоичное информационное слово i длиной 4. Получить кодовое слово c по формуле циклического кодирования.
3. Внести в кодовое слово c произвольную ошибку (инвертировать любой бит слова). Вычислить синдром. Локализовать ошибку и исправить еѐ. Получить информационное слово. Убедиться в идентичности полученного информационного и изначального слов.
4. Создать подпрограмму для реализации алгоритма помехоустойчивого циклического кодирования для кода(7;4). На вход подпрограмме передаѐтся информационное слово. Подпрограмма возвращает кодовое слово.
5. Создать подпрограмму для декодирования кодового слова с учѐтом возможной ошибки. На вход подпрограммы передаѐтся кодовое слово. Подпрограмма вычисляет синдром, и при наличии ошибки исправляет еѐ. Далее подпрограмма выделяет информационные биты и возвращает информационное слово.
6. Создать программу, демонстрирующую работу подпрограмм. Программа позволяет пользователю ввести информационное слово. Далее вызывается первая подпрограмма, и слово кодируется, затем заносится или не заносится случайная ошибка. Далее программа вызывает вторую подпрограмму и декодирует кодовое слово, исправляя ошибку. Результат каждого этапа выводится на экран. Примечание: при ручном счѐте и написании программы реализовывать процессы кодирования слова и вычисления синдрома путѐм операций над многочленами в поле GF(2).

**Выполнение**

**Задание 1**

|  |  |  |
| --- | --- | --- |
| **+ 1** | (x + 1) \* (+ x + 1) \* (+ + 1) | Множители |
| **g(x)** | ++1 | Порождающий многочлен |
| **h(x)** | **=** x4+x2+x+1. | Проверочный многочлен |

**Задание 2**

I = 1101

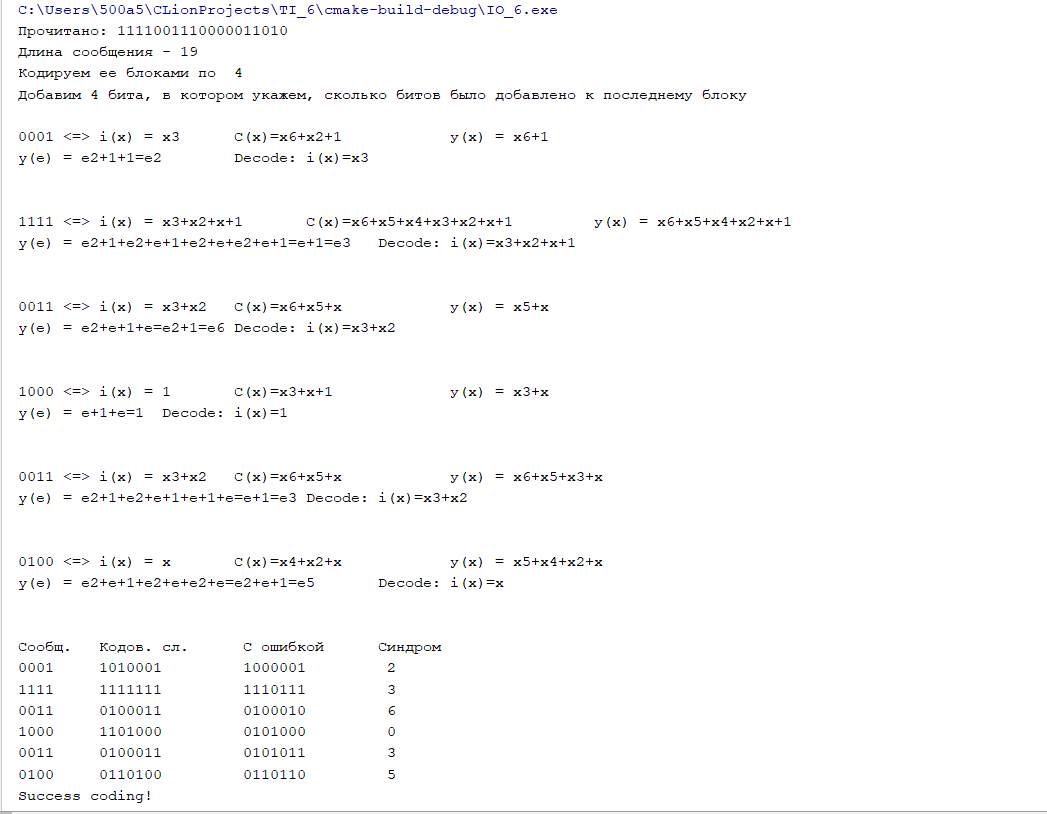
c(x) = \* i(x) + Rg(x) \* ( \* i(x))

Rg(x) = (xr \* i(x)) ***mod*** g(x)

кодовое слово с= ++

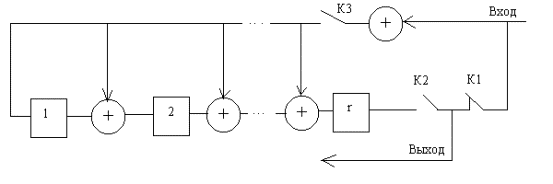
**Задание 3**

Добавляем в C(x) ошибку => +++В случае одиночной ошибки на позиции *i* полином ошибок равен *e(x) = xi*. Синдром слова *y(x)* равен:*S(y)=y(ɛ) = c(ɛ) + e(ɛ) = e(ɛ) = ɛi* **y(e)=+1++e+e+1+=   
S = 0010 = 2  
c(x)=**++ **i(x)=+x+1**



**Ответы на вопросы**

1. *Определение конечного поля. Арифметические операции в конечном поле.*«**Поле** — множество, для элементов которого определены операции сложения, вычитания, умножения и деления (кроме деления на нуль), причём свойства этих операций близки к свойствам обычных числовых операций». **Конечное** **поле**, или **поле** Галуа в общей алгебре — **поле**, состоящее из **конечного** числа элементов.
2. *Поле GF(2).* Многочлены над полем GF(2). Арифметические операции над многочленами.  
   Довольно часто при построении помехоустойчивых кодов используются многочлены над полем GF(2),коэффициенты которых складывались и умножа-лись по правилам сложения и умножения элементов GF(2). В то же время все приведенные выше операции справедливы для многочленов над полем GF(2n),которое является расширением поля GF(2)
   1. Сложение (х5 + х4 + х2 + х) + (х3 + х2 + х + 1) = х5 + х4 + х3 + 12.
   2. Вычитание совершается аналогично.
   3. Умножение полиномов производится по ***mod***(хn+1), т.е. за результат берется остаток от деления результата обычного умножения полиномов на двучлен хn+1
3. Неприводимые многочлены   
   Многочлен p(x1, x2, …, xn) p ( x 1 , x 2 , . . , x n ) {\displaystyle p(x\_{1},x\_{2},..,x\_{n})} от n n {\displaystyle n} переменных над полем k k {\displaystyle k} называется неприводимым над полем k {\displaystyle k} k, если он является простым элементом кольца k [ x 1 , x 2 , . . , x n ] {\displaystyle k[x\_{1},x\_{2},..,x\_{n}]} k(x1, x2, …, xn), то есть не является константой и не представим в виде произведения p = q r {\displaystyle p=qr} p = q\*r, где q q {\displaystyle q} и r r {\displaystyle r} ― многочлены с коэффициентами из k {\displaystyle k} k, отличные от констант.
4. Рассматривается кольцо многочленов K[x] K [ x ] {\displaystyle K[x]} над полем K K {\displaystyle K} . Два многочлена g1 g 1 {\displaystyle g\_{1}} и g2 g 2 {\displaystyle g\_{2}} , принадлежащие выбранному кольцу, называются **сравнимыми** по модулю многочлена f {\displaystyle f} f, если их разность g1 – g2 g 1 − g 2 {\displaystyle g\_{1}-g\_{2}} делится на f f {\displaystyle f} без остатка. Сравнение обозначается следующим образом: g1 == g2 (mod f)  
   Так же, как и в кольце целых чисел, такие сравнения можно складывать, вычитать и перемножать
5. определение примитивного циклического кода   
   Циклический код — линейный, блочный код, обладающий свойством цикличности, то есть каждая циклическая перестановка кодового слова также является кодовым словом. Используется для преобразования информации для защиты её от ошибок
6. Ключи к1 и к2 первоначально замкнуты, а ключ к3 – разомкнут. Исходная комбинация через ключ к1 поступает на выход и через входной сумматор на сдвиговый регистр, где и образуется контрольные символы. Затем ключ к2 замыкается, а к1 и к3 размыкаются. Контрольные символы подаются на выход в след, за информационными символами.



1. определение порождающего многочлена циклического кода  
   Нормированный ненулевой многочлен g(x) наименьшей степени из множества многочленов циклического кода называется порождающим многочленом циклического кода.
2. определение проверочного многочлена циклического кода  
   Пусть С – циклический код с порождающим многочленом g(x) и длиной кодового слова равной n , тогда проверочным многочленом кода С называется многочлен вида: https://konspekta.net/studopediaorg/baza4/1493258765698.files/image628.gif .
3. Связь порождающего и проверочного многочленов одного и того же кода.  
   Если порождающий многочлен циклического кода g[x] делит без остатка многочлен x^n, то x^n = h(x) \* g(x)  
   h(x) – проверочный многочлен
4. Алгоритм циклического кодирования. Систематическое кодирование  
   На практике чаще всего применяется алгоритм кодирования, который формирует *систематический разделимый* код. В основу такого алгоритма положена *операция деления*на ***G****(x).* Систематические разделимые коды привлекательны тем, что процедуру кодирования, т.е. преобразования информационного вектора ***A***(длины *k* ) в вектор кода ***V***(длины *n>k* ) удается свести лишь к формированию (*n-k* ) контрольных бит.  
    Шаг 1. Предварительно вектор ***A***«отнормируем по формату» под длину n, воспользовавшись операцией умножения многочленов ***A****(x)* *×* *xn-k*. Как было показано в лекционном курсе – это эквивалентно сдвигу вектора ***A***на (*n-k* ) позиций влево. Произведение многочленов на языке векторов имеет длину n. Существенно для последующего, что правые (*n-k* ) позиций оказываются *непременно нулевыми.*  
    Шаг 2. Произведение ***A****(* *x* *)* *×* *xn - k*разделим на ***G****(* *x* *).* Ясно, что в общем случае оно не обязано делиться на ***G****(* *x* *)* нацело. Поэтому следует записать ***A****(x)* *×* *xn-k =****Q****(x)* *×* ***G****(x)+****R****(x),*  
   где ***Q****(* *x* *)* - частное от деления; ***R****(* *x* *)* - остаток.   
   Это многочлен степени не больше (*n* *-* *k* *‑1* ), т. к. делитель имеет степень (*n* *-* *k* ) по определению. Как вектор он имеет длину (*n* *-* *k* ).  
    Шаг 3. Перенесём остаток ***R****(* *x* *)* в левую часть равенства. Получим:  
   ***A****(x)* *×* *xn-k +****R****(x)=****Q****(x)* *×* ***G****(x* ).  
   Теперь в левой части мы получаем многочлен, который *нацело делится на****G****(x)* , а это по определению – многочлен, принадлежащий циклическому (*n, k* ) – коду. В этой последней операции остаток ***R***складывается с нулями (см. шаг1 алгоритма). Следовательно, конечный итог эквивалентен конкатенированию ***R***к вектору ***А***.  
     
   **Систематическое кодирование** заключается в сопоставлении информационному вектору (x_1,\dots,x_k) кодового слова (c_0,c_1,\dots,c_{n-1}) по правилу:c_0+c_1x+\dots+c_{n-1}x^{n-1}\equiv (x_1+x_2x+\dots+x_kx^{k-1}) x^{n-k} - R(x) \ ,  
   где R(x) — остаток от деления полинома (x_1+x_2x+\dots+x_kx^{k-1}) x^{n-k} на порождающий код полином g_{}(x).
5. Алгоритм локализации ошибок в циклическом коде.
   1. Принятая комбинация делится на образующий многочлен g(x). Если остаток R(x)<>0 то определяется вес остатка w. Если вес остатка равен или меньше числа исправляемых ошибок t (w<=t), то принятую комбинацию складывают по модулю 2 с остатком и получают исправленную комбинацию.
   2. Если w>t, то производится циклический сдвиг на один символ влево и полученная после такого сдвига комбинация снова делится на образующий многочлен. Если вес полученного остатка w<=t, то циклически сдвинутую комбинацию складывают с остатком и затем после сложения циклически сдвигают в обратную сторону вправо на один символ (возвращают на прежнее место). В результате получаем исправленную комбинацию.
   3. Если после циклического сдвига на один символ по прежнему w>t, то производят дополнительные циклические сдвиги влево. При этом после каждого сдвига осуществляется деление сдвинутой комбинации на g(x) и проверяется вес остатка. При w<=t сдвинутую комбинацию складывают с остатком и производят обратных циклических сдвигов вправо столько, сколько было сделано влево.