Построение циклических кодов

**§ 1 Введение**

Код ,в котором кодовая комбинация, полученная путем циклического сдвига разрешенной кодовой комбинации является также разрешенной кодовой комбинацией называется циклическим ( полиномиальным, кодом с циклическими избыточными проверками-ЦИП).

Сдвиг осуществляется справа налево, при этом крайний левый символ переносится в конец комбинации.

Циклический код относится к линейным, блочным, корректирующим, равномерным кодам.

В циклических кодах кодовые комбинации представляются в виде многочленов, что позволяет свести действия над кодовыми комбинациями к действием над многочленами (используя аппарат полиномиальной алгебры).

Циклические коды являются разновидностью систематических кодов и поэтому обладают всеми их свойствами. Первоначально они были созданы для упрощения схем кодирования и декодирования. Их эффективность при обнаружении и исправлении ошибок обеспечила им широкое применение на практике.

Циклические коды используются в ЭВМ при последовательной передаче данных .

**2 Постановка задачи**

Построить циклический код для передачи 31 разрядной кодовой комбинации с исправлением однократной ошибки ( n=31 ,s=1) двумя

способами.

Показать процесс обнаружения и исправления однократной ошибки в передаваемой кодовой комбинации. Составить программу, реализующую алгоритм кодирования, декодирования и исправления ошибки при передаче данных с использованием циклического кода.

**3 Операции над циклическими кодами**

1. Сдвиг справа налево осуществляется путем умножения полинома на x:

G(x)=x4+x2+1 ⇔ 0010101;

G(x)⋅x=x5+x3+x ⇔ 0101010.

2. Операции сложения и вычитания выполняются по модулю 2 .

Они являются эквивалентными и ассоциативными :

G1(x)+G2(x)=>G3(x);

G1(x) -G2(x)=>G3(x);

G2(x)+G1(x)=>G3(x);

Пример:

G1(x)=x5 +x3+x;

G2(x)=x4 +x3 +1;

G3(x)=G1(x) ⊕ G2(x) = x5 +x4+x+1.

3. Операция деления является обычным делением многочленов, только вместо вычитания используется сложеное по модулю 2 :

G1(x)=x6+x4+x3 ;

G2(x)=x3+x2+1 .

**4 Принцип построения циклических кодов**

Идея построения циклических кодов базируется на использовании неприводимых многочленов. Неприводимым называется многочлен, который не может быть представлен в виде произведения многочленов низших степеней ,т.е. такой многочлен делиться только на самого себя или на единицу и не делиться ни на какой другой многочлен. На такой многочлен делиться без остатка двучлен xn+1.Неприводимые многочлены в теории циклических кодов играют роль образующих полиномов.

Чтобы понять принцип построения циклического кода, умножаем комбинацию простого k-значного кода Q(x) на одночлен xr ,а затем делим на образующий полином P(x) , степень которого равна r. В результате умножения Q(x) на xr степень каждого одночлена, входящего в Q(x), повышается на r. При делении произведения xrQ(x) на образующий полином получается частное C(x) такой же степени, как и Q(x).

Частное C(x) имеет такую же степень, как и кодовая комбинация Q(x) простого кода, поэтому C(x) является кодовой комбинацией этого же простого k-значного кода. Следует заметить, что степень остатка не может быть больше степени образующего полинома, т.е. его наивысшая степень может быть равна (r-1). Следовательно, наибольшее число разрядов остатка R(x) не превышает числа r.

Умножая обе части равенства (1) на P(x) и произведя некоторые перестановки получаем :

F(x) = C(x) P(x) = Q(x) xr + R(x) (2)

Таким образом, кодовая комбинация циклического n-значного кода может

быть получена двумя способами:

1) умножение кодовой комбинации Q(x) простого кода на одночлен xr

и добавление к этому произведению остатка R(x) , полученного в результате деления произведения Q(x) xr на образующий полином P(x);

2) умножения кодовой комбинации C(x) простого k-значного на образующий полином P(x).

При построении циклических кодов первым способом расположение информационных символов во всех комбинациях строго упорядочено - они занимают k старших разрядов комбинации, а остальные (n-k) разрядов отводятся под контрольные.

При втором способе образования циклических кодов информационные и контрольные символы в комбинациях циклического кода не отделены друг от друга, что затрудняет процесс декодирования.

**6. Разработка текста программы**

Для представления информационного слова в памяти используется

массив. В состав программы входит основная программа и два модуля,

реализующие алгоритм кодирования и декодирования информационных слов и диалога с пользователем соответственно.

Program Cyclic\_Code;

Uses

Crt,\_CC31,\_Serv;

Var

m,mm:Move\_code;

p:Polinom;

r:Rest;

i,Mainflag,From,Error:integer;

Switch:byte;

Key:boolean;

begin

Repeat

Key:=true;

TextColor(11);

TextBackGround(7);

Clrscr;

SetWindow(24,10,45,14,2,' Главное меню ');

Switch:=GetMainMenuChoice;

case Switch of

1:begin

About;

Readln;

Key:=False;

end;

2: begin

TextColor(0);

ClrScr;

SetWindow(25,10,40,13,1,' Образовать ');

Switch:=GetSubMenuChoice;

case Switch of

1:begin

TextBackGround(0);

TextColor(15);

ClrScr;

SetWindow(1,1,79,24,2,' Демонстрация');

TextColor(14);

GotoXY(2,2);

Init(m,p,r,MainFlag);

Write(‘Информационный полином ');

TextColor(2);

for i:=n downto 0 do

begin

if(i<n-n1+1)then Textcolor(9);

Write(m[i]);

end;

TextColor(14);

GotoXY(2,3);

Write('Образующий полином ');

TextColor(13);

for i:=n1 downto 0 do

Write(p[i]);

TextColor(14);

GotoXY(2,4);

Write('Сложение по модулю 2 (F(x)+P(x)): ');

FxPx(m);

TextColor(9);

for i:=n downto 0 do

begin

if(i<n1)then TextColor(2);

Write(m[i]);

end;

TextColor(14);

GotoXY(2,5);

Write('Остаток: ');

Divizion(m,r,p,Mainflag);

TextColor(11);

for i:=n1 downto Mainflag do

Write(r[i]);

GotoXY(2,6);

TextColor(14);

Write('Передаваемый полином: ');

BildMoveCode(m,r,Mainflag);

TextColor(9);

for i:=n downto 0 do

begin

if(i<n1) then TextColor(11);

Write(m[i]);

end;

GotoXY(2,7);

TextColor(14);

Write('Произошла ошибка... ');

MakeError(m,Error);

TextColor(9);

for i:=n downto 0 do

begin

if(i=Error)then

TextColor(12)

else

TextColor(9);

write(m[i]);

end;

GotoXY(2,8);

TextColor(14);

Write('Ошибка исправлена! ');

TextColor(9);

Correction(m,p,r);

for i:=n downto 0 do

begin

if(i=Error)then

TextColor(10)

else

TextColor(9);

write(m[i]);

end;

TextColor(14);

GotoXY(2,9);

Write('Исходный полином: ');

Decoder(m);

TextColor(2);

for i:=n downto 0 do

begin

if(i<n-n1+1)then Textcolor(9);

Write(m[i]);

end;

Key:=false;

end;

2:begin

TextBackGround(0);

TextColor(15);

ClrScr;

SetWindow(1,1,79,24,2,'Демонстрация');

TextColor(14);

GotoXY(2,2);

Init(m,p,r,MainFlag);

Write('Информационный полином: ');

TextColor(2);

for i:=n downto 0 do

begin

if(i<n-n1+1)then Textcolor(9);

Write(m[i]);

end;

TextColor(14);

GotoXY(2,3);

Write('Образующий полином: ');

TextColor(13);

for i:=n1 downto 0 do

Write(p[i]);

TextColor(14);

GotoXY(2,4);

Write('Результат умножения: ');

BildMoveCodeMultiplication(m);

TextColor(9);

for i:=n downto 0 do

Write(m[i]);

GotoXY(2,5);

TextColor(14);

Write('Произошла ошибка ... ');

MakeError(m,Error);

TextColor(9);

for i:=n downto 0 do

begin

if(i=Error)then

TextColor(12)

else

TextColor(9);

write(m[i]);

end;

GotoXY(2,6);

TextColor(14);

Write('Ошибка исправлена ! ');

TextColor(9);

Correction(m,p,r);

for i:=n downto 0 do

begin

if(i=Error)then

TextColor(10)

else

TextColor(9);

write(m[i]);

end;

Key:=false;

end;

end;

TextColor(14);

GotoXY(2,22);

Write('Нажмите любую клавишу...');

Readln;

end;

3:begin

ClrScr;

GotoXY(1,24);

TextColor(14);

Writeln('Работа программы завершена ...');

Readln;

TextBackGround(0);

TextColor(15);

ClrScr;

Key:=true;

end;

end;

Until Key;

end.

**7 .Результаты работы программы**

Результат работы программы при образовании кода добавлением остатка

Демонстрация

Информационный полином: 0000011010111110011110110110110

Образующий полином: 111101

Cложениe по модулю 2 (F(x)+P(x)): 1101011111001111011011011000000

Остаток: 010101

Передаваемый полином: 1101011111001111011011011010101

Произошла ошибка... 1101011111001110011011011010101

Ошибка исправлена! 1101011111001111011011011010101

Исходный полином: 0000011010111110011110110110110

Нажмите любую клавишу...

Результат работы при образовании кода умножением

Демонстрация

Информационный полином: 0000001010110000011111010001011

Образующий полином: 111101

Результат умножения: 0110000011111010000100100101111

Произошла ошибка... 0110000011111010000100100101101

Ошибка исправлена! 0110000011111010000100100101111

Нажмите любую клавишу...

Выводы:

Данная программа кодирует сообщения используя циклический код.

При этом она имитирует работу канала для передачи информации.

При возникновении исключительных ситуаций, когда информационное слово по каким-либо причинам раскодировать не удаётся, программа повторяет запрос на пересылку данных, как это делается в реальных ситуациях подобного рода.

Кроме этого, программа случайным образом, "при прохождении

информационного слова через канал" допускает в слове однократную ошибку, затем исправляет ее, декодирует информационное слово и передаёт результат пользователю.

**Приложение № 1**

Процедуры и функции модуля \_сс31.

Unit \_CC31;

Interface

Uses

Crt;

Const

n=30; { Информация+код }

n1=5; { Размер контрольных разрядов }

Type

Move\_code=array[0..n] of byte; { Передаваемый полином F(x) }

Rest=array[0..n1] of byte; { Остаток }

Polinom=array[0..n1] of byte; { Образующий полином P(x) }

Procedure Init(var m1:Move\_code;var p1:Polinom;

var r1:Rest;var flag:integer);

Procedure FxPx(var m6:Move\_Code);

Procedure Divizion(var m2:Move\_code;var r2:Rest;

p2:Polinom;var flag:integer);

Procedure BildMoveCode(var m3:Move\_code;r3:Rest;var flag:integer);

Procedure Decoder(var m6:Move\_Code);

Procedure MakeError(var m4:Move\_code;var err:integer);

Procedure BildMoveCodeMultiplication(var m7:Move\_Code);

Procedure Correction(var m5:Move\_code;p5:Polinom;var r5:Rest);

Implementation

Procedure Init;

var

i:integer;

begin

p1[5]:=1;

p1[4]:=1;

p1[3]:=1;

p1[2]:=1;

p1[1]:=0;

p1[0]:=1;

flag:=0;

for i:=n1 downto 0 do

r1[i]:=0;

Randomize;

for i:=n-n1 downto 0 do

m1[i]:=random(2);

end;

Procedure FxPx(var m6:Move\_Code);

var

i:integer;

k:byte;

begin

k:=5;

while(k>0) do

begin

for i:=n downto 1 do

m6[i]:=m6[i-1];

dec(k);

end;

for i:=n1-1 downto 0 do

m6[i]:=0;

end;

Procedure Divizion(var m2:Move\_code;var r2:Rest;

p2:Polinom;var flag:integer);

label

RETURN;

var

i,j,i1,kol,Countzero:integer;

begin

j:=n;

RETURN:while((j>=0)and(m2[j]=0))do dec(j);

if(j>n1)

then begin

for i:=n1 downto 0 do

begin

r2[i]:=m2[j];

dec(j);

end;

while(j>=0)do

begin

for i:=n1 downto 0 do

r2[i]:=r2[i] xor p2[i];

i1:=n1;

while((i1>=0)and(r2[i1]=0))do dec(i1);

if(i1=-1)then goto RETURN;

Kol:=n1-i1;

while(Kol>0)do

begin

for i:=n1 downto 1 do

r2[i]:=r2[i-1];

dec(Kol);

end;

Kol:=n1-i1;

while((Kol>0)and(j>=0))do

begin

r2[Kol-1]:=m2[j];

dec(Kol);

dec(j);

end;

if((j=-1)and(Kol=0))

then begin

for i:=n1 downto 0 do

r2[i]:=r2[i] xor p2[i];

end

else flag:=Kol;

end;

end

else if(n1=j)

then begin

for i:=n1 downto 0 do

begin

r2[i]:=m2[j];

dec(j);

end;

for i:=n1 downto 0 do

r2[i]:=r2[i] xor p2[i]

end

else if(j<n1)

then begin

for i:=j downto 0 do

r2[i]:=m2[i]

end;

end;

Procedure BildMoveCode(var m3:Move\_code;r3:Rest;var flag:integer);

var

i,k:integer;

begin

if(flag>0)then

begin

k:=n1-flag;

for i:=n1 downto flag do

begin

m3[k]:=r3[i];

dec(k);

end;

end

else begin

for i:=n1-1 downto 0 do

m3[i]:=r3[i];

end;

end;

Procedure MakeError(var m4:Move\_code;var err:integer);

begin

Randomize;

err:=Random(n);

m4[err]:=m4[err] xor 1;

end;

Procedure Decoder(var m6:Move\_Code);

var

i:integer;

k:byte;

begin

k:=5;

while(k>0) do

begin

for i:=0 to n-1 do

m6[i]:=m6[i+1];

dec(k);

end;

for i:=n downto n-n1+1 do

m6[i]:=0;

end;

Procedure BildMoveCodeMultiplication(var m7:Move\_Code);

var

m1,m2,m3,m4,mm:Move\_Code;

i,j:integer;

begin

mm:=m7;

m1:=m7;

for j:=0 to 1 do

begin

for i:=n downto 1 do

m1[i]:=m1[i-1];

m1[j]:=0;

end;

m2:=m7;

for j:=0 to 2 do

begin

for i:=n downto 1 do

m2[i]:=m2[i-1];

m2[j]:=0;

end;

m3:=m7;

for j:=0 to 3 do

begin

for i:=n downto 1 do

m3[i]:=m3[i-1];

m3[j]:=0;

end;

m4:=m7;

for j:=0 to 4 do

begin

for i:=n downto 1 do

m4[i]:=m4[i-1];

m4[j]:=0;

end;

for i:=n downto 0 do

m7[i]:=mm[i] xor m1[i]xor m2[i]xor m3[i] xor m4[i];

end;

Procedure Correction(var m5:Move\_code;p5:Polinom;var r5:Rest);

var

i,Correctflag,i1:integer;

Count,Countcarry,Carryflag:byte;

begin

Correctflag:=0;

Countcarry:=0;

repeat

for i:=n1 downto 0 do

r5[i]:=0;

Count:=0;

Divizion(m5,r5,p5,Correctflag);

i1:=n1;

while((i1>=Correctflag)and(r5[i1]=0))do dec(i1);

if({(i1=Correctflag-1) or

(}(i1=Correctflag)and(r5[Correctflag]=1)){)}

then m5[0]:=m5[0] xor r5[Correctflag]

else begin

Carryflag:=m5[n];

for i:=n downto 1 do

m5[i]:=m5[i-1];

m5[0]:=Carryflag;

inc(Countcarry);

end;

until ({(i1=Correctflag-1) or

(}(i1=Correctflag)and(r5[Correctflag]=1));{);}

while (Countcarry>0) do

begin

Carryflag:=m5[0];

for i:=0 to n-1 do

m5[i]:=m5[i+1];

m5[n]:=Carryflag;

dec(Countcarry);

end;

end;

end.

**Приложение № 2**

Процедуры и функции модуля \_Serv.

Unit \_SERV;

Interface

Uses

Crt,Dos;

Const

EmptyBorder =0;

SingleBorder =1;

DoubleBorder =2;

BorderChar:array[0..2,1..6] of Char=

((#32,#32,#32,#32,#32,#32),

(#218,#196,#191,#179,#192,#217),

(#201,#205,#187,#186,#200,#188));

MaxChar=80;

MaxLine=25;

MenuTop=3;

SubMenuTop =2;

MenuLine :array[1..MenuTop]of string[20]=

(' О программе...',' Демонстрация ' ‘Выход ');

SubMenuLine :array[1..SubMenuTop]of string[20]=

(' Сложением' , ' Умножением');

Procedure SetWindow(x1,y1,x2,y2,Bord:byte;Header:string);

Procedure CursorOff;

Function GetMainMenuChoice:byte;

Function GetSubMenuChoice:byte;

Procedure About;

Implementation

Procedure SetWindow(x1,y1,x2,y2,Bord:byte;Header:string);

var

i:integer;

begin

if not ((x1<1) or (x2<=x1) or

(y1<1) or (y2<=y1) or (x2>MaxChar) or

(y2>MaxLine) or (Bord>2)) then

begin

GotoXY(x1,y1);

Write(BorderChar[Bord,1]);

for i:=1 to x2-x1-1 do

begin

GotoXY(x1+i,y1);

Write(BorderChar[Bord,2]);

end;

GotoXY(x2,y1);

Write(BorderChar[Bord,3]);

for i:=1 to y2-y1-1 do

begin

GotoXY(x1,y1+i);

Write(BorderChar[Bord,4]);

GotoXY(x2,y1+i);

Write(BorderChar[Bord,4]);

end;

GotoXY(x1,y2);

Write(BorderChar[Bord,5]);

for i:=1 to x2-x1-1 do

begin

GotoXY(x1+i,y2);

Write(BorderChar[Bord,2]);

end;

GotoXY(x2,y2);

Write(BorderChar[Bord,6]);

end;

GotoXY((x2-x1-ord(Header[0])) div 2+x1,y1);

Write(Header)

end;

Procedure CursorOff;

begin

asm

mov ah,1

mov ch,20h

int 10h

end;

end;

Function GetMainMenuChoice:byte;

var

Count:byte;

i:integer;

ch,ch1:char;

begin

Count:=1;

while KeyPressed do

ch:=Readkey;

repeat

for i:=1 to MenuTop do

begin

if(i=Count)then

begin

HighVideo;

TextColor(0);

end

else

begin

LowVideo;

TextColor(8);

end;

GotoXY(25,10+i);

Writeln(MenuLine[i]);

CursorOff;

end;

if KeyPressed

then begin

ch:=Readkey;

if(ch=#0)

then begin

ch1:=Readkey;

case ch1 of

#72 : if(Count>1)

then dec(Count);

#80 : if(Count<MenuTop)

then inc(Count);

end;

end;

end;

until(ch=#13);

GetMainMenuChoice:=Count;

end;

Function GetSubMenuChoice:byte;

var

Count:byte;

i:integer;

ch,ch1:char;

begin

Count:=1;

while KeyPressed do

ch:=Readkey;

repeat

for i:=1 to SubMenuTop do

begin

if(i=Count)then

begin

HighVideo;

TextColor(9);

end

else

begin

LowVideo;

TextColor(1);

end;

GotoXY(26,10+i);

Writeln(SubMenuLine[i]);

CursorOff;

end;

if KeyPressed

then begin

ch:=Readkey;

if(ch=#0)

then begin

ch1:=Readkey;

case ch1 of

#72 : if(Count>1)

then dec(Count);

#80 : if(Count<SubMenuTop)

then inc(Count);

end;

end;

end;

until(ch=#13);

GetSubMenuChoice:=Count;

end;

Procedure About;

begin

TextColor(15);

SetWindow(5,1,75,3,1,'О программе');

TextColor(10);

GotoXY(6,2);

TextColor(10+128);

Write('Токарь Алексей Юрьевич АП-57.Курсовой проект.

“Циклический код” ');

end;

end.