Министерство образования и науки Украины

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовому проекту

на тему "Кодирование информации. Код Рида-Малера "

по курсу "Кодирование и защита информации"

2003

Содержание

Введение

1 Избыточные коды

2 Алгоритм кодирования Рида-Малера

3 Тестовый пример

Вывод

Библиографический список

Приложение: Текст программы

Введение

Целью настоящей работы является закрепление знаний, получаемых в процессе изучения дисциплины, приобретение необходимых практических навыков применения методов кодирования информации. В данной курсовой работе рассматривается кодирование информации методом Рида-Малера. Код Рида-Малера относится к избыточным кодам. Заданием на данную работу было разработать алгоритм и программу кодирования и декодирования данных кодом Рида-Малера (16,11).

1. Избыточные коды

Избыточные коды – одно из наиболее эффективных средств обеспечения высокой достоверности передаваемых и принимаемых сообщений.

Избыточные коды могут применяться с целью либо только обнаружения возможных ошибок, либо исправления обнаруженных ошибок. Во всех случаях желательно достичь максимальной корректирующей способности. Но в зависимости от конкретного построения кода его способность к исправлению тех или иных ошибок может изменяться в широких пределах.

При некоторых значениях n и k может быть найдено такое удачное построение кода, при котором веса всех разрешенных ненулевых комбинаций мало отличаются друг от друга и от половины максимально возможного веса. При других значениях n и k может оказаться, что для некоторой малой части кодовых комбинаций из их общего числа расстояние оказывается существенно меньшим, чем для большинства других. Поэтому при рассмотрении характеристик кодов можно обнаружить, что близкие по избыточности и числу разрядов коды резко отличаются друг от друга по своей корректирующей способности.

В кодах с одной проверкой на четность содержится всего один проверочный символ. Этот символ выбирается таким, чтобы его сумма по модулю два со всеми информационными символами равнялась нулю. Благодаря такому способу выбора проверочного символа кодовая комбинация содержит четное число единиц. И следовательно признаком искажения кодовой комбинации является нечетность единиц в принятой комбинации. Данный код позволяет только обнаруживать однократные ошибки и все ошибки нечетной кратности.

В кодах с простым повторением положе метод повторения исходной кодовой комбинации. На приемной стороне комбинация складывается с исходной и при нулевой сумме комбинация принимается. Этот код позволяет обнаруживать ошибки, за исключением ошибок в парных элементах.

Корелляционный код. Удваиваются символы, при этом если в разряде информационной части стоит 0, то он заменяется на 01, а 1 – на 10. Сигналом ошибки является появление 00 или 11.

В инверсном коде используется повторение исходной комбинации следующим образом: если комбинация содержит нечетное число единиц, то вместо 1 ставится 0, а всесто 0 – 1. Если четное число единиц, то она удваивается без инверсии. На приемной стороне подсчитывается число единиц и, если оно четно, то вторая половинка инвертируется и складывается с первой. Если же число единиц четно, то вторая складывается с первой и должен получиться 0.

2 Алгоритм кодирования Рида-Малера

Коды Рида-Малера – это блоковые коды, которые строятся следующим образом:

 (1)  (2)  (3),

n-длина блока

K-число информационных символов

d-минимальное кодовое расстояние

m-положительное, условное число не меньше 3

σ-порядок кода, который всегда меньше, чем m.

Т.е. в зависимости от порядка при одном и том же m можно получить разные коды.

m=4; σ=1,2,3; n=24=16.

Построение кодов Рида-Маллера сводится к следующему.

В начале строится производящая матрица G, первая строка которой содержит n единиц. Далее следует m строк, совокупность которых удобно рассматривать как (m\*n) –матрицу, в качестве столбцов которой выбраны двоичные числа (начиная с нуля). Номера разрядов двоичных чисел удобно считать сверху вниз. Эти m строк определяют векторы первого порядка δ. Далее идут строки векторов второго порядка, которые получаются из всех произведений двух строк первого порядка, затем - строки третьего порядка, являющиеся всеми произведениями трех строк первого порядка и т д.

Пример: т=3 σ=2 п=2т=23=8

 



Для кодирования определяется общее число символов в блоке через информационные символы, суммируя ненулевые позиции соответствующего столбика, образующей матрицы. Единицы в столбцах матрицы G показывают, какие именно информационные символы Uk определяют значение символов Ui кодового слова.



Пусть пришла последовательность:



Получаем: 11101011

Декодирование осуществляется по мажоритарному принципу или принципу большинства.

Декодирование осуществляется вначале для всех информационных символов (кроме 1-го) на основе так называемых парных компонентов. Начинать запись таких уравнений надо с векторов максимального порядка.

В нашем примере σ=2=> первым выписывается Uk5.

0 0 0 1 0 0 0 1

Для векторов 2-го порядка парными считаются компоненты:

00 → 0

01 → 1

На втором уровне сочетаний каждый 0 соединяется с каждой 1 попарно. Теперь в проверяемое равенство выписываются все объединенные позиции 1-го и 2-го уровней.





Вычисляем символ Uk6

0 0 0 0 0 1 0 1



Для Uk7:



При декодировании с помощью векторов 1-го порядка мы также точно пользуемся парными компонентами, но поскольку здесь 1-ый уровень, то мы объединяем просто 0 и 1, стоящую на соответствующих позициях, и, во-вторых, при декодировании в полученных уравнениях используют не исходное, а преобразованное уравнение, которое получается путем сложения по модулю два исходного уравнения и векторов 2-го порядка, (потому что матрица имеет 2-ой порядок),



0 1 0 1 0 1 0 1



0 0 1 1 0 0 1 1



0 0 0 0 1 1 1 1



После этого еще раз преобразуют исходное выражение: к полученному преобразованному выражению прибавляем векторы 1-го порядка, которые содержат единицу в соответствующем информационном разряде.



Если в полученном выражении получили все 1, то значит Uk1=1, а если все 0, то Uk1=0.

Если при передаче произошли искажения, то вычисляемый символ по каждой системе проверок выбирается по мажоритарному принципу, в том числе и для символа Uk1.

3 Тестовый пример

В качестве тестового примера использовалась комбинация 11001010000.

В задании к курсовому проекту указано, что n=16, а k=11. Таким образом получили образующую матрицу, которая имеет следующий вид:



Построение матрицы реализовано программно для заданных n и k.

После кодирования исходной комбинации 11001010000 по полученной образующей матрице получили комбинацию1010111101010000. При этом мы суммируем те разряды исходной комбинации, на соответствующих позициях которых, в рассматриваемом столбце образующей матрицы, стоят единицы. Например, возьмем четвертый столбец. Единицы имеются в разрядах 1,2,3,6. При суммировании этих разрядов получим 0. То есть в закодированной комбинации в четвертом разряде получили 0:

Декодирование осуществляется по мажоритарному признаку, то есть при декодировании каждый разряд комбинации проверяется несколькими уравнениями. Благодаря этому можно избежать ошибок при передаче. Даже если в каком-либо разряде ошибка, она поменяет результат только одного уравнения. А в остальных результат останется тот же.

**Вывод**

В ходе курсовой работы был исследован простой помехоустойчивый код, а именно код Рида-Малера. Данный код обнаруживает ошибки, но не исправляет их. Рассматривался случай, когда n=16, k=11, поэтому в матрице присутствуют только вектора первого и второго порядка. Однако если появится необходимость использовать также вектора третьего порядка, то образующая матрица станет значительно больше, что не выгодно с точки зрения экономии памяти. При этом также увеличится длина закодированной комбинации, что увеличит время передачи сообщения по каналу связи.

Библиографический список

1. Березюк Н.Т. «Кодирование Информации» - 1978.
2. Конспект лекций по дисциплине «Кодирование и защита информации».

Приложение

Текст программы

//---------------------------------------------------------------------------

#include <vcl.h>

#pragma hdrstop

#include "Unit1.h"

//---------------------------------------------------------------------------

#pragma package(smart\_init)

#pragma resource "\*.dfm"

TForm1 \*Form1;

const N=16;

const K=11;

char \*s;

int f,l[11],l1[16];

//---------------------------------------------------------------------------

\_\_fastcall TForm1::TForm1(TComponent\* Owner)

: TForm(Owner)

{int i,j;

for (i=0;i<N;i++)

{ StringGrid1->Cells[i][0]=1; }

for (i=0;i<N;i++)

{ StringGrid1->Cells[i][1]=0;

i++; StringGrid1->Cells[i][1]=1; }

for (i=0;i<N;i++)

{ StringGrid1->Cells[i][2]=0;

i++; StringGrid1->Cells[i][2]=0;

i++; StringGrid1->Cells[i][2]=1;

i++; StringGrid1->Cells[i][2]=1; }

for (i=0;i<N;i++)

{ if (i<N/2)

{ StringGrid1->Cells[i][4]=0; }

else

{ StringGrid1->Cells[i][4]=1; }

}

for (i=0;i<N;i++)

{ if (7<i&&i<12||4>i&&i<8)

{ StringGrid1->Cells[i][3]=0; }

else { StringGrid1->Cells[i][3]=1; }

}

int b;

int k1,k2,a=5;

j=1; b=1;

for (j;j<4;j++)

{ for(i=0;i<N;i++)

{ k1=StringGrid1->Cells[i][b].ToInt();

j++; k2=StringGrid1->Cells[i][j].ToInt(); j--;

if (k1==1&&k2==1)

{ StringGrid1->Cells[i][a]=1; }

else

{ StringGrid1->Cells[i][a]=0; }

}

a++; }

b=2; j=2;

for (j;j<4;j++)

{ for(i=0;i<N;i++)

{ k1=StringGrid1->Cells[i][b].ToInt(); j++;

k2=StringGrid1->Cells[i][j].ToInt(); j--;

if (k1==1&&k2==1)

{ StringGrid1->Cells[i][a]=1; }

else

{ StringGrid1->Cells[i][a]=0; }

}

a++;

}

b=3; j=3;

for (j;j<4;j++)

{ for(i=0;i<N;i++)

{ k1=StringGrid1->Cells[i][b].ToInt(); j++;

k2=StringGrid1->Cells[i][j].ToInt(); j--;

if (k1==1&&k2==1)

{ StringGrid1->Cells[i][a]=1; }

else

{ StringGrid1->Cells[i][a]=0; }

}

}

s=" "; Edit1->Text=s;

Edit1->MaxLength=12;

Memo1->Text=s;

Memo1->Lines->Add("U1=Uk1");

Memo1->Lines->Add("U2=Uk1+Uk2");

Memo1->Lines->Add("U3=Uk1+Uk3");

Memo1->Lines->Add("U4=Uk1+Uk2+Uk3+Uk6");

Memo1->Lines->Add("U5=Uk1+Uk4");

Memo1->Lines->Add("U6=Uk1+Uk2+Uk4+Uk7");

Memo1->Lines->Add("U7=Uk1+Uk3+Uk4+Uk9");

Memo1->Lines->Add("U8=Uk1+Uk2+Uk3+Uk4+Uk6+Uk7+Uk9");

Memo1->Lines->Add("U9=Uk1+Uk5");

Memo1->Lines->Add("U10=Uk1+Uk2+Uk5+Uk8");

Memo1->Lines->Add("U11=Uk1+Uk3+Uk5+Uk10");

Memo1->Lines->Add("U12=Uk1+Uk2+Uk3+Uk5+Uk6+Uk8+Uk10");

Memo1->Lines->Add("U13=Uk1+Uk4+Uk5+Uk11");

Memo1->Lines->Add("U14=Uk1+Uk2+Uk4+Uk5+Uk7+Uk8+Uk11");

Memo1->Lines->Add("U15=Uk1+Uk3+Uk4+Uk5+Uk9+Uk10+Uk11");

Memo1->Lines->Add("U16=Uk1+Uk2+Uk3+Uk4+Uk5+Uk6+Uk7+Uk8+Uk9+Uk10+Uk11");}

//---------------------------------------------------------------------------

void \_\_fastcall TForm1::BitBtn1Click(TObject \*Sender)

{s=Edit1->Text.c\_str();

for(int i=1;i<K+1;i++)

{ l[i-1]=StrToInt(s[i]); }

for (int i=0;i<N;i++)

{ f=0;

for (int j=0;j<K;j++)

{ if (StringGrid1->Cells[i][j]==1)

{ f=l[j]+f; if (f==2) f=0;

}

l1[i]=f;

}

}

for(int i=0;i<N;i++)

{ StringGrid2->Cells[i][0]=l1[i];

}

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm1::Button2Click(TObject \*Sender)

{ int ed, nul,i,j,perem[4];

int l2[16]; int edin1=0,edin2=0,edin3=0,edin4=0,edin5=0,edin6=0;

ed=0; nul=0;

for(int i=0;i<N;i++)

{ l1[i]=StringGrid2->Cells[i][0].ToInt(); }

for(i=0;i<N;i++)

{ l2[i]=l1[i]; }

perem[0]=l1[0]+l1[4]+l1[8]+l1[12]; perem[1]=l1[1]+l1[5]+l1[9]+l1[13];

perem[2]=l1[2]+l1[6]+l1[10]+l1[14];perem[3]=l1[3]+l1[7]+l1[11]+l1[15];

for (i=0;i<4;i++)

{ if (perem[i]==1||perem[i]==3)

{ perem[i]=1; ++ed; }

else

{ perem[i]=0; ++nul; }

}

if (nul<ed)

{ StringGrid3->Cells[10][0]=1; edin1=10; }

else

{ StringGrid3->Cells[10][0]=0; }

//------------------------------

ed=0; nul=0;

perem[0]=l1[0]+l1[2]+l1[8]+l1[10]; perem[1]=l1[1]+l1[3]+l1[9]+l1[11];

perem[2]=l1[4]+l1[6]+l1[12]+l1[14]; perem[3]=l1[5]+l1[7]+l1[13]+l1[15];

for (i=0;i<4;i++)

{ if (perem[i]==1||perem[i]==3)

{ perem[i]=1; ++ed; }

else

{ perem[i]=0; ++nul; }

}

if (nul<ed)

{ StringGrid3->Cells[9][0]=1; edin2=9; }

else

{ StringGrid3->Cells[9][0]=0; }

//----------------------------

ed=0; nul=0;

perem[0]=l1[0]+l1[2]+l1[4]+l1[6]; perem[1]=l1[1]+l1[3]+l1[5]+l1[7];

perem[2]=l1[8]+l1[10]+l1[12]+l1[14]; perem[3]=l1[9]+l1[11]+l1[13]+l1[15];

for (i=0;i<4;i++)

{ if (perem[i]==1||perem[i]==3)

{ perem[i]=1; ++ed; }

else

{ perem[i]=0; ++nul; }

}

if (nul<ed)

{ StringGrid3->Cells[8][0]=1; edin3=8; }

else

{ StringGrid3->Cells[8][0]=0; }

//----------------------------

ed=0; nul=0;

perem[0]=l1[0]+l1[1]+l1[8]+l1[9]; perem[1]=l1[2]+l1[3]+l1[10]+l1[11];

perem[2]=l1[4]+l1[5]+l1[12]+l1[13]; perem[3]=l1[6]+l1[7]+l1[14]+l1[15];

for (i=0;i<4;i++)

{ if (perem[i]==1||perem[i]==3)

{ perem[i]=1; ++ed; }

else

{ perem[i]=0; ++nul; }

}

if (nul<ed)

{ StringGrid3->Cells[7][0]=1; edin4=7; }

else

{ StringGrid3->Cells[7][0]=0; }

//---------------------------

ed=0; nul=0;

perem[0]=l1[0]+l1[1]+l1[4]+l1[5]; perem[1]=l1[2]+l1[3]+l1[6]+l1[7];

perem[2]=l1[8]+l1[9]+l1[12]+l1[13]; perem[3]=l1[10]+l1[11]+l1[14]+l1[15];

for (i=0;i<4;i++)

{ if (perem[i]==1||perem[i]==3)

{ perem[i]=1; ++ed; }

else

{ perem[i]=0; ++nul; }

}

if (nul<ed)

{ StringGrid3->Cells[6][0]=1; edin5=6; }

else

{ StringGrid3->Cells[6][0]=0; }

//----------------------------

ed=0; nul=0;

perem[0]=l1[0]+l1[1]+l1[2]+l1[3]; perem[1]=l1[4]+l1[5]+l1[6]+l1[7];

perem[2]=l1[8]+l1[9]+l1[10]+l1[11]; perem[3]=l1[12]+l1[13]+l1[14]+l1[15];

for (i=0;i<4;i++)

{ if (perem[i]==1||perem[i]==3)

{ perem[i]=1; ++ed; }

else

{ perem[i]=0; ++nul; }

}

if (nul<ed)

{ StringGrid3->Cells[5][0]=1; edin6=5; }

else

{ StringGrid3->Cells[5][0]=0; }

//ПРЕОБРАЗОВАНHOЕ СООБЩЕНИЯ

if (edin1!=0)

for (i=0;i<N;i++)

{ l1[i]=l1[i]+StringGrid1->Cells[i][10].ToInt(); }

if (edin2!=0)

for (i=0;i<N;i++)

{ l1[i]=l1[i]+StringGrid1->Cells[i][9].ToInt(); }

if (edin3!=0)

for (i=0;i<N;i++)

{ l1[i]=l1[i]+StringGrid1->Cells[i][8].ToInt(); }

if (edin4!=0)

for (i=0;i<N;i++)

{ l1[i]=l1[i]+StringGrid1->Cells[i][7].ToInt(); }

if (edin5!=0)

for (i=0;i<N;i++)

{ l1[i]=l1[i]+StringGrid1->Cells[i][6].ToInt(); }

if (edin6!=0)

for (i=0;i<N;i++)

{ l1[i]=l1[i]+StringGrid1->Cells[i][5].ToInt(); }

for(i=0;i<N;i++)

{ if (l1[i]==1||l1[i]==3||l1[i]==5||l1[i]==7)

{ l1[i]=1; }

else

{ l1[i]=0; }

}

//-----------------------------------

int edin7=0,edin8=0,edin9=0,edin10=0;

ed=0;nul=0;

int perem1[8];

perem1[0]=l1[0]+l1[8]; perem1[1]=l1[1]+l1[9];

perem1[2]=l1[2]+l1[10]; perem1[3]=l1[3]+l1[11];

perem1[4]=l1[4]+l1[12]; perem1[5]=l1[5]+l1[13];

perem1[6]=l1[6]+l1[14]; perem1[7]=l1[7]+l1[15];

for (i=0;i<4;i++)

{ if (perem1[i]==1||perem1[i]==3||perem1[i]==5||perem1[i]==7)

{ perem1[i]=1; ++ed; }

else

{ perem1[i]=0; ++nul; }

}

if (nul<ed)

{ StringGrid3->Cells[4][0]=1; edin7=6; }

else

{ StringGrid3->Cells[4][0]=0; }

//------------------------------

ed=0; nul=0;

perem1[0]=l1[0]+l1[4]; perem1[1]=l1[1]+l1[5];

perem1[2]=l1[2]+l1[6]; perem1[3]=l1[3]+l1[7];

perem1[4]=l1[8]+l1[12]; perem1[5]=l1[9]+l1[13];

perem1[6]=l1[10]+l1[14]; perem1[7]=l1[11]+l1[15];

for (i=0;i<4;i++)

{ if (perem1[i]==1||perem1[i]==3||perem1[i]==5||perem1[i]==7)

{ perem1[i]=1; ++ed; }

else

{ perem1[i]=0; ++nul; }

}

if (nul<ed)

{ StringGrid3->Cells[3][0]=1; edin8=7; }

else

{ StringGrid3->Cells[3][0]=0; }

//----------------------------------

ed=0; nul=0;

perem1[0]=l1[0]+l1[2]; perem1[1]=l1[1]+l1[3];

perem1[2]=l1[4]+l1[6]; perem1[3]=l1[5]+l1[7];

perem1[4]=l1[8]+l1[10]; perem1[5]=l1[9]+l1[11];

perem1[6]=l1[12]+l1[14]; perem1[7]=l1[13]+l1[15];

for (i=0;i<4;i++)

{ if (perem1[i]==1||perem1[i]==3||perem1[i]==5||perem1[i]==7)

{ perem1[i]=1; ++ed; }

else

{ perem1[i]=0; ++nul; }

}

if (nul<ed)

{ StringGrid3->Cells[2][0]=1; edin9=8; }

else

{ StringGrid3->Cells[2][0]=0; }

//----------------------------

ed=0; nul=0;

perem1[0]=l1[0]+l1[1]; perem1[1]=l1[2]+l1[3];

perem1[2]=l1[4]+l1[5]; perem1[3]=l1[6]+l1[7];

perem1[4]=l1[8]+l1[9]; perem1[5]=l1[10]+l1[11];

perem1[6]=l1[12]+l1[13]; perem1[7]=l1[14]+l1[15];

for (i=0;i<4;i++)

{ if (perem1[i]==1||perem1[i]==3||perem1[i]==5||perem1[i]==7)

{ perem1[i]=1; ++ed; }

else

{ perem1[i]=0; ++nul; }

}

if (nul<ed)

{ StringGrid3->Cells[1][0]=1; edin10=9; }

else

{ StringGrid3->Cells[1][0]=0; }

//2-e преобразование

int l3[11];

if (edin7!=0)

for (i=0;i<N;i++)

{ l1[i]=l1[i]+StringGrid1->Cells[i][4].ToInt(); }

if (edin8!=0)

for (i=0;i<N;i++)

{ l1[i]=l1[i]+StringGrid1->Cells[i][3].ToInt(); }

if (edin9!=0)

for (i=0;i<N;i++)

{ l1[i]=l1[i]+StringGrid1->Cells[i][2].ToInt(); }

if (edin10!=0)

for (i=0;i<N;i++)

{ l1[i]=l1[i]+StringGrid1->Cells[i][1].ToInt(); }

for(i=0;i<N;i++)

{ if (l1[i]==1||l1[i]==3||l1[i]==5)

{ l1[i]=1; }

else

{ l1[i]=0; }

}

//-------------------------------

int perem2=0;

for(i=0;i<N;i++)

{ if (l1[i]==1)

{ ++ed; }

else

{ ++nul; }

}

if (nul<ed)

{ StringGrid3->Cells[0][0]=1; }

else

{ StringGrid3->Cells[0][0]=0; }

}

//---------------------------------------------------------------------------