

C.N Lab 2

Title:

Implementation of Hamming Code for error detection.

Lohith Kandula, AP20110010161, lohith_ranganadhareddy@srmap.edu.in ,
CSE C

Submitted on: 29 August 2022

Objective:

While Sender is sending a message to Receiver through a channel, it might get corrupted. The receiver has to check whether the received message is corrupted or not without having an original copy. So, we need an error detection using Hamming Code to detect up to 2 simultaneous bit errors and correct single-bit errors.

Problem Statement:

The message might get corrupted due to noise in the channel. Here, the communication channel corrupts only one bit. Hence, the receiver might either receive the correct message or the incorrect message. Hamming code is used to solve the error detection and error correction for single-bit errors only. If all the parity bits are 0's then there is no error detected in the transmitted message i.e code word. If all the parity bits are not 0's then there is an error detected in the transmitted message which prints the position of the single bit error and displays the corrected code word.

Algorithm:

- Calculation of the number of redundant bits
- Finding the position of parity/redundant bits
- Filling the parity bits
- Sending the message on the receiver side, the steps are:
- Receiving the message
- Finding the position of parity/redundant bits
- Comparing the parity bits to find the error

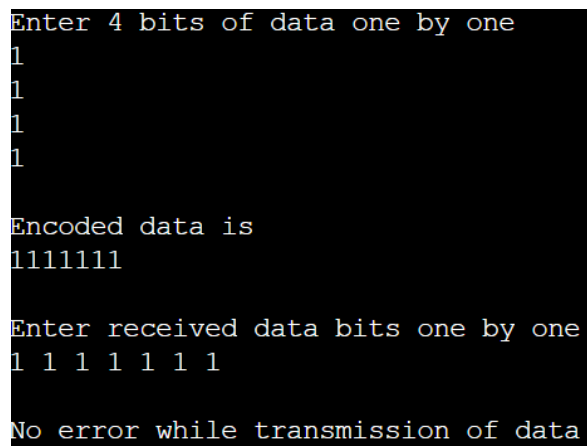
CODE:

```
#include<iostream>
using namespace std;
int main() {
    int data[10];
    int dataatrec[10],c,c1,c2,c3,i;
    cout<<"Enter 4 bits of data one by one\n";
    cin>>data[0];
    cin>>data[1];
    cin>>data[2];
    cin>>data[4];
    //Calculation of even parity
    data[6]=data[0]^data[2]^data[4];
    data[5]=data[0]^data[1]^data[4];
    data[3]=data[0]^data[1]^data[2];
    cout<<"\nEncoded data is\n";
    for(i=0;i<7;i++)
        cout<<data[i];
    cout<<"\n\nEnter received data bits one by one\n";
    for(i=0;i<7;i++)
        cin>>dataatrec[i];
    c1=dataatrec[6]^dataatrec[4]^dataatrec[2]^dataatrec[0];
    c2=dataatrec[5]^dataatrec[4]^dataatrec[1]^dataatrec[0];
    c3=dataatrec[3]^dataatrec[2]^dataatrec[1]^dataatrec[0];
    c=c3*4+c2*2+c1 ;
    if(c==0) {
        cout<<"\nNo error while transmission of data\n";
    }
    else {
        cout<<"\nError on position "<<c;
        cout<<"\nData sent : ";
        for(i=0;i<7;i++)
            cout<<data[i];
        cout<<"\nData received : ";
        for(i=0;i<7;i++)
            cout<<dataatrec[i];
        cout<<"\nCorrect message is\n";
        //if erroneous bit is 0 we complement it else vice versa
```

```
if(dataatrec[7-c]==0)
dataatrec[7-c]=1;
    else
dataatrec[7-c]=0;
for (i=0;i<7;i++) {
cout<<dataatrec[i];
}
}
return 0;
}
```

OUTPUT:

TEST CASE-1: without error



```
Enter 4 bits of data one by one
1
1
1
1
1

Encoded data is
1111111

Enter received data bits one by one
1 1 1 1 1 1 1

No error while transmission of data
```

Explanation

Here we gave input as '1111' and encoded data as '1111111' we gave the received data without any error so the computer displays that the data is transmitted without any error.

TEST CASE-2: with error

```
Enter 4 bits of data one by one
1
0
0
1

Encoded data is
1001100

Enter received data bits one by one
1 0 1 1 1 0 0

Error on position 5
Data sent : 1001100
Data received : 1011100
Correct message is
1001100
```

Explanation

Here we gave input as '1001' and encoded data as '1011100' we gave the received data with an error so the computer displays the error that occurred on position 5 and it shows the data sent and data received. And it shows the correct message as '1001100'

PROBLEMS FACED:

Starting I wasn't able to understand the Hamming Code algorithm and wasn't able to implement the code. I felt difficult in understanding the usage of the extra parity bits

CONCLUSION:

I have understood the concept and logic used in the Hamming Code and implemented using C++ language. My coding skills in the c++ language have improved. I was able to detect errors in the received message.