Byte stuffing binary version

#include <stdio.h>

#include <string.h>

#define S 100

char flag[] = {'0', '1', '1', '1', '1', '1', '1', '0'};

char esc[] = {'0', '0', '0', '1', '1', '0', '1', '1'};

void reciever(char data[])

{

    char res[S];

    int count = 0, count1 = 0, j = 0, k = 0, l = 0, p = 0;

    while (1)

    {

        int isFlag = 0, isEsc = 0;

        for (int i = 0; i < 8; i++)

        {

            if (data[j + i] == flag[i])

            {

                isFlag = 1;

            }

            else

            {

                isFlag = 0;

                break;

            }

        }

        for (int m = 0; m < 8; m++)

        {

            if (data[j + m] == esc[m])

            {

                isEsc = 1;

            }

            else

            {

                isEsc = 0;

                break;

            }

        }

        if (isFlag == 1 && count1 == 0)

        {

            count++;

            l++;

        }

        else if (isEsc == 1 && count1 == 0)

        {

            count1++;

            l++;

        }

        else

        {

            for (int i = 0; i < 8; i++)

            {

                res[p++] = data[j + i];

            }

            count1 = 0;

        }

        j += 8;

        if (count == 2)

        {

            break;

        }

    }

    res[strlen(data) - l \* 8] = '\0';

    printf("Original Data is %s", res);

}

int main()

{

    char data[S];

    char data1[S];

    char data2[S];

    printf("Enter Data:");

    scanf("%s", data);

    data[strlen(data)] = '\0';

    // covert bits to bytes by appending leading zero's

    int c\_bytes;

    if (strlen(data) % 8 == 0)

    {

        c\_bytes = strlen(data) / 8;

        for (int i = 0; i < c\_bytes \* 8; i++)

        {

            data1[i] = data[i];

        }

        data1[c\_bytes \* 8] = '\0';

    }

    else

    {

        c\_bytes = (strlen(data) / 8) + 1;

        for (int i = 0; i < (c\_bytes \* 8) - strlen(data); i++)

        {

            data1[i] = '0';

        }

        int j = 0;

        for (int i = (c\_bytes \* 8) - strlen(data); i < c\_bytes \* 8; i++)

        {

            data1[i] = data[j++];

        }

        data1[c\_bytes \* 8] = '\0';

    }

    printf("Data is of %d bytes.\n", c\_bytes);

    printf("Data Converted in Bytes format: %s\n", data1);

    // append esc if data bits are equal to flag or esc key

    int j = 0, k = 0, count1 = 0;

    int count = c\_bytes;

    while (1)

    {

        int isFlag = 1, isEsc = 1;

        for (int i = 0; i < 8; i++)

        {

            if (data1[j + i] != flag[i])

            {

                isFlag = 0;

                break;

            }

        }

        for (int m = 0; m < 8; m++)

        {

            if (data1[j + m] != esc[m])

            {

                isEsc = 0;

                break;

            }

        }

        if (isFlag == 1 || isEsc == 1)

        {

            count1++;

            for (int i = 0; i < 8; i++)

            {

                data2[k++] = esc[i];

            }

        }

        for (int l = 0; l < 8; l++)

        {

            data2[k++] = data1[j++];

        }

        count--;

        if (count == 0)

        {

            break;

        }

    }

    data2[strlen(data1) + count1 \* 8] = '\0';

    printf("Data after inserting Esc key if data having flag or Esc byte: %s\n", data2);

    // create final data by appending flag at both ends

    for (int i = 0; i < 8; i++)

    {

        data1[i] = flag[i];

    }

    for (int i = 0; i < strlen(data2); i++)

    {

        data1[8 + i] = data2[i];

    }

    for (int i = 0; i < 8; i++)

    {

        data1[8 + strlen(data2) + i] = flag[i];

    }

    data1[16 + strlen(data2)] = '\0';

    printf("Final data to be sent: %s\n", data1);

    reciever(data1);

    return 0;

}

Byte stuffing

#include <stdio.h>

#include <string.h>

void main()

{

    char frame[50][50], str[50][50];

    char flag[10];

    strcpy(flag, "flag");

    char esc[10];

    strcpy(esc, "esc");

    int i, j, k = 0, n;

    strcpy(frame[k++], "flag");

    printf("Enter length of String : \n");

    scanf("%d", &n);

    printf("Enter the String: ");

    for (i = 0; i <= n; i++)

    {

        gets(str[i]);

    }

    printf("\nYou entered :\n");

    for (i = 0; i <= n; i++)

    {

        puts(str[i]);

    }

    printf("\n");

    for (i = 1; i <= n; i++)

    {

        if (strcmp(str[i], flag) != 0 && strcmp(str[i], esc) != 0)

        {

            strcpy(frame[k++], str[i]);

        }

        else

        {

            strcpy(frame[k++], "esc");

            strcpy(frame[k++], str[i]);

        }

    }

    strcpy(frame[k++], "flag");

    // printf("------------------------------\n\n");

    printf("Byte stuffing at sender side:\n\n");

    // printf("------------------------------\n\n");

    for (i = 0; i < k; i++)

    {

        printf("%s\t", frame[i]);

    }

}

Bit stuffing

#include <stdio.h>

#include <string.h>

char flag[] = {'0', '1', '1', '1', '1', '1', '1', '0'};

#define s 50

void receiver(char data[])

{

    char res[s];

    int z, j, k = 0;

    int count = 0;

    for (int i = 0; i < strlen(data); i++)

    {

        count = 0;

        if (data[i] == '0')

        {

            count++;

            for (j = i + 1; j < i + 8; j++)

            {

                if (data[j] == '1')

                {

                    count++;

                }

                else

                {

                    break;

                }

            }

        }

        if (count == 7)

        {

            i = j;

        }

        else

        {

            res[k++] = data[i];

        }

    }

    res[strlen(data) - 16] = '\0';

    printf("Data after Flag bytes dropped by reciever: %s\n", res);

    k = 0;

    int count1 = 0;

    for (int i = 0; i < strlen(res); i++)

    {

        count = 0;

        if (res[i] == '0')

        {

            data[k++] = res[i];

            for (int j = i + 1; j < i + 7; j++)

            {

                if (res[j] == '1')

                {

                    data[k++] = res[j];

                    count++;

                }

                else

                {

                    break;

                }

            }

        }

        if (count == 5)

        {

            i = j;

            count1++;

        }

        else

        {

            data[k++] = res[i];

        }

    }

    data[strlen(res) - count1] = '\0';

    printf("Orignal Data recieved: %s\n", data);

}

int main()

{

    char data[s];

    char data1[s];

    printf("Enter the data: ");

    scanf("%s", &data);

    data[strlen(data)] = '\0';

    // after zero --continous five 1's stuff zero.

    int count = 0;

    int count1 = 0;

    int k = 0;

    for (int i = 0; i < strlen(data); i++)

    {

        data1[k++] = data[i];

        if (data[i] == '0')

        {

            count = 0;

            for (int j = i + 1; j < i + 6; j++)

            {

                data1[k++] = data[j];

                if (data[j] == '1')

                {

                    count++;

                    if (count == 5)

                    {

                        count1++;

                        data1[k++] = '0';

                    }

                }

                else

                {

                    i = j;

                    break;

                }

                i = j;

            }

        }

    }

    data1[strlen(data) + count1] = '\0';

    printf("Data after stuffing 0's: %s \n", data1);

    // flag.

    for (int i = 0; i < strlen(data1) + 16; i++)

    {

        if (i < 8)

        {

            data[i] = flag[i];

        }

        else if (i > strlen(data1) + 7)

        {

            data[i] = flag[i - strlen(data1) - 8];

        }

        else

        {

            data[i] = data1[i - 8];

        }

    }

    data[strlen(data1) + 16] = '\0';

    printf("Final Data to be sent: %s\n", data);

    receiver(data);

}

CRC  
  
#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#define N strlen(crc\_key)

void sender();

void receiver();

void error();

void crc(char data[]);

// Declerations

char data[50];    // Char Array for data

char crc\_key[20]; // Char array for crc key

int data\_len;

int i, j;

char chk\_val[50]; // Stores the remainder during the CRC function.

int ch;

void crc(char data[])

{

    char remainder[N + 1]; // To store the remainder during CRC calculations.

    for (i = 0; i < N; i++)

        chk\_val[i] = data[i];

    i = N;

    do

    {

        if (chk\_val[0] == '1')

        {

            // XOR with CRC key.

            for (j = 1; j < N; j++)

                chk\_val[j] = ((chk\_val[j] == crc\_key[j]) ? '0' : '1');

        }

        // Shifting to left

        for (j = 0; j < N - 1; j++)

        {

            chk\_val[j] = chk\_val[j + 1];

        }

        chk\_val[j] = data[i++];

        printf("Remainder%d: %s\n", i - N, chk\_val);

    } while (i <= data\_len + N - 1);

}

void sender()

{

    printf("\nEnter data : ");

    scanf("%s", data);

    data\_len = strlen(data); // Storing the length of initial data.

    printf("\nEnter the crc key: ");

    scanf("%s", crc\_key);

    for (i = data\_len; i < (data\_len + N - 1); i++)

    {

        data[i] = '0';

    }

    data[data\_len + N - 1] = '\0';

    printf("\nAppended data with N-1 0s: %s", data);

    printf("\n\n");

    // Performing CRC function

    crc(data);

    printf("\n\nCRC or Check value is : %s", chk\_val);

    for (i = data\_len; i < data\_len + N - 1; i++)

    {

        data[i] = chk\_val[i - data\_len];

    }

    printf("\nFinal data to be sent : %s", data);

    printf("\n");

}

void receiver()

{

    char receivedData[50];

    strcpy(receivedData, data);

    printf("\nChecking data\n");

    crc(receivedData);

    // Checking weather the final remainder is all '0' or not if '1' found error detected.

    for (int k = 0; k < strlen(crc\_key); k++)

    {

        if (chk\_val[k] == '1')

        {

            printf("\nError detected. Data corrupted.\n");

            return; // No need to go further just return

        }

    }

    printf("\nData received: %s", receivedData);

}

void error()

{

    int errorPos;

    printf("\nEnter position to add error : ");

    scanf("%d", &errorPos);

    data[errorPos] = (data[errorPos] == '1') ? '0' : '1';

}

int main()

{

    do

    {

        printf("\n\n1) Without error\n2) With error\n3) Exit");

        printf("\nEnter your choice:");

        scanf("%d", &ch);

        switch (ch)

        {

        case 1:

        {

            sender();

            receiver();

            break;

        }

        case 2:

        {

            sender();

            error();

            receiver();

            break;

        }

        case 3:

        {

            exit(0);

            break;

        }

        default:

        {

            break;

        }

        }

    } while (ch < 4 && ch > 0);

    return 0;

}

Dsss

#include <stdio.h>

#include <string.h>

#define S 110

char \*org\_signl(char data[], char data1[]);

char \*XNOR(char data1[], char result[]);

char \*org\_dataa(char org\_result[], char org\_data[]);

int main(){

char data[10];

char data1[S] = {'\0'};

char result[S] = {'\0'};

char org\_result[S] = {'\0'};

char org\_data[10] = {'\0'};

printf("Enter Data: ");

scanf("%s",data);

org\_signl(data,data1);

printf("%s\n",data1);

XNOR(data1,result);

printf("Result: %s\n",result);

XNOR(result,org\_result);

printf("Org\_result: %s\n",org\_result);

org\_dataa(org\_result,org\_data);

printf("Org\_Data: %s\n",org\_data);

return 0;

}

char \*org\_signl(char data[], char data1[]){

int k=0;

for(int i=0; i<strlen(data); i++){

if(data[i]=='1'){

for(int j=0; j<11; j++){

data1[k++] = '1';

}

}else{

for(int j=0; j<11; j++){

data1[k++] = '0';

}

}

}

data1[strlen(data)\*11]='\0';

return data1;

}

char \*XNOR(char data1[], char result[]){

char pattern[] = {'1','0','1','1','0','1','1','1','0','0','0'};

int l = strlen(data1);

int k=0;

for(int i = 0; i < l; i += 11){

for(int j = 0; j < 11; j++){

if(data1[i+j] == pattern[j]){

result[k++]='1';

}else{

result[k++]='0';

}

}

}

result[k] = '\0';

return result;

}

char \*org\_dataa(char org\_result[], char org\_data[]){

int flag = -1, j = 0;

int l = strlen(org\_result);

for(int i = 0; i < l; i += 11){

if(org\_result[i] == '1'){

flag = 1;

}else if(org\_result[i] == '0'){

flag = 0;

}else{

flag = -1;

printf("Something Wrong\n");

break;

}

org\_data[j++] = flag + '0';

}

org\_data[j]='\0';

return org\_data;

}

Hamming for flexible number of bits

#include <stdio.h>

#include <math.h>

int calculateParityBits(int data\_bits) {

    int m = 0;

    while (pow(2, m) < (data\_bits + m + 1))

        m++;

    return m;

}

void generateHammingCode(int data[], int data\_bits) {

    int m = calculateParityBits(data\_bits);

    int hamming\_code[data\_bits + m];

    int i, j, k;

    // Initialize Hamming code array with 0

    for (i = 0; i < data\_bits + m; i++)

        hamming\_code[i] = 0;

    // Copy data bits into Hamming code array skipping parity bit positions

    j = k = 0;

    for (i = 1; i <= data\_bits + m; i++) {

        if (i == pow(2, k)) {

            k++;

        } else {

            hamming\_code[i] = data[j++];

        }

    }

    // Calculate parity bits

    for (i = 0; i < m; i++) {

        int parity\_position = pow(2, i);

        int count = 0;

        for (j = 1; j <= data\_bits + m; j++) {

            if (j & parity\_position) {

                if (hamming\_code[j] == 1)

                    count++;

            }

        }

        if (count % 2 == 0)

            hamming\_code[parity\_position] = 0;

        else

            hamming\_code[parity\_position] = 1;

    }

    printf("Hamming code: ");

    for (i = 1; i <= data\_bits + m; i++)

        printf("%d", hamming\_code[i]);

    printf("\nNumber of parity bits: %d\n", m);

}

int main() {

    int data[1000]; // Assuming maximum 1000 bits

    int i, data\_bits;

    printf("Enter number of data bits (maximum 1000): ");

    scanf("%d", &data\_bits);

    if (data\_bits <= 0 || data\_bits > 1000) {

        printf("Invalid number of data bits.\n");

        return 1;

    }

    printf("Enter data bits (each separated by a space): ");

    for (i = 0; i < data\_bits; i++)

        scanf("%1d", &data[i]);

    generateHammingCode(data, data\_bits);

    return 0;

}

Stop and wait

#include <stdio.h>

#include <stdlib.h>

int numofPacket;

float tp;

float tt;

float time = 0;

int i, j, k;

void sender();

void receiver();

void sender(){

    printf("\nEnter number of packets :");

    scanf("%d", &numofPacket);

    int packet[numofPacket];

    printf("\nEnter packets : ");

    for(i = 0; i < numofPacket; i++){

        scanf("\n%d", &packet[i]);

    }

    printf("\nEnter transmission time in ms : ");

    scanf("%f", &tt);

    printf("\nEnter propagation time in ms : ");

    scanf("%f", &tp);

    printf("\nTheoretically time taken to transmit %d number of packets should be %.2f", numofPacket, numofPacket\*2\*(tt+tp));

    printf("\n");

    for(i = 0; i < numofPacket; i++){

        time += tt;

        printf("\nFor %d : \n",packet[i]);

        printf("\nBuffer time is %.2f",time);

        receiver();

    }

    printf("\n");

    printf("\nTotal time taken is %.2f", time);

    printf("\n");

}

void receiver(){

    time += tp;

    printf("\nPacket received at %.2f", time);

    time += tt;

    printf("\nAcknowledgement sent at %.2f", time);

    time += tp;

    printf("\nAcknowledgement received at %.2f", time);

    printf("\n-----------------------");

    printf("\n");

}

int main(){

    sender();

    return 0;

}