Telekomunikacja - laboratorium	Studia stacjonarne - inzynierskie

Nazwa zadania	Kody wykrywające i korygujące błędy transmisji

Dzień	poniedziałek	Godzina	14:00- 15:30	Rok akademicki	2020/2021

Imię i Nazwisko	Kamil Budzyn - 229850
Imię i Nazwisko	Wiktor Bechciński - 229840

Opis programu, rozwiązania problemu.

Program przekodowuje dowolny plik do postaci zakodowanej i odkodowuje go do postaci pierwotnej z korekcją do dwóch błędów powstałych w trakcie transmisji. Macierz która została użyta w programie posiada osiem bitów parzystości. Musi ona spełniać warunki: -brak kolumny zerowej, -wszystkie kolumny są różne, -.żadna kolumna nie jest sumą dwóch pozostałych.

Macierz H:

```
{{1, 0, 1, 0, 0, 1, 0, 1,  1, 0, 0, 0, 0, 0, 0, 0},
 {0, 1, 1, 1, 0, 1, 1, 1,  0, 1, 0, 0, 0, 0, 0, 0},
 {0, 0, 0, 1, 1, 1, 1, 0,  0, 0, 1, 0, 0, 0, 0, 0},
 {1, 0, 0, 0, 1, 1, 1, 1,  0, 0, 0, 1, 0, 0, 0, 0},
 {1, 1, 1, 0, 0, 0, 1, 0,  0, 0, 0, 0, 1, 0, 0, 0},
 {1, 1, 1, 1, 0, 0, 0, 1,  0, 0, 0, 0, 0, 1, 0, 0},
 {1, 1, 0, 1, 1, 1, 0, 1,  0, 0, 0, 0, 0, 0, 1, 0},
 {0, 1, 0, 0, 1, 0, 1, 1,  0, 0, 0, 0, 0, 0, 0, 1} }
```

Najważniejsze elementy kodu programu z opisem.

```
// FUNKCJA KODUJĄCA WIADOMOŚĆ Z PLIKU
getline(wejscie, linia);                        //WCZYTANIE DO ZMIENNEJ 'LINIA' PIERWSZEJ A NASTEPNIE KOLEJNYCH LINII PLIKU
for (int k = 0; k < linia.length(); k++)
        {
           int litera = int(convertToASCII(linia.substr(k, 1)));  //POBRANIE PIERWSZEJ A NASTEPNIE KOLEJNYCH LITER Z DANEJ
                                                                  // LINII

           for (int i = BYTE - 1; i >= 0; i--)                    //KONWERSJA Z ASCII NA SYSTEM BINARNY
           {
              litera_bin[i] = litera % 2;
              litera /= 2;
           }
           for (int i = 0; i < BYTE; i++)
           {
              kontrola[i] = 0; // WYPELNIENIE TABLICY KONTROLNEJ ZERAMI

              for (int j = 0; j < BYTE; j++)
              {
                 kontrola[i] += litera_bin[j] * H[i][j]; // LICZENIE KOLEJNYCH BITOW PARZYSTOSCI
              }
              kontrola[i] %= 2;
           }

           for (int i = 0; i < BYTE; i++)              // ZAPIS ODCZYTANEGO ZNAKU DO PLIKU
           {
              zakodowane << litera_bin[i];
           }
           for (int i = 0; i < BYTE; i++)
           {
              zakodowane << kontrola[i];
           }
           zakodowane << endl;



(...)



        getline(zakodowane, linia);          //WCZYTANIE DO ZMIENNEJ 'LINIA' PIERWSZEJ A NASTEPNIE KOLEJNYCH LINII PLIKU
        do
        {
           blad1 = 0;
           for (int i = 0; i < linia.length(); i++)
           {
```

```
    zakodowane_tab[i] = int(convertToASCII(linia.substr(i, 1))) - 48;  //KONWERSJA Z ZAPISU BINARNEGO NA
                                                                       //ZMIENNE INT
}

for (int i = 0; i < BYTE; i++)
{
   blad_tab[i] = 0;          //ZEROWANIE TABLICY BLEDOW
   for (int j = 0; j < BYTE * 2; j++)
   {
      blad_tab[i] += zakodowane_tab[j] * H[i][j];    //MNOZENIE MACIERZY T ORAZ H JEŻELI T*H!=0 OZNACZA TO
                                                     //ŻE W ZAKODOWANEJ WIADOMOŚCI ZNAJDUJE SIĘ BŁĄD
   }
   blad_tab[i] %= 2;
   if (blad_tab[i] == 1) {
      blad1 = 1;            //BLAD ZOSTAL ZNALEZIONY
   }
}

if (blad1 != 0)
{              //JEZELI BLAD2==0 TO JEST JEDEN BLAD, JEZELI BLAD2==1 TO SA 2 BLEDY
   blad2 = 0;
   for (int i = 0; i < BYTE * 2; i++)
   {                  //SPRAWDZENIE ILE JEST BLEDOW - 1 CZY 2
      for (int j = i + 1; j < BYTE * 2; j++)
      {
         blad2 = 1;
         for (int k = 0; k < BYTE; k++)
         {
            if (blad_tab[k] != H[k][i] ^ H[k][j])
            {
               blad2 = 0; //XOR NA WYBRANYCH KOLUMNACH MACIERZY H - SPRAWDZENIE CZY SA 2 BLEDY
               break;
            }
         }

         if (blad2 == 1)                    //PRZYPADEK DLA 2 BLEDOW
         {
            firstError = i;
            secondError = j;
            zakodowane_tab[firstError] = !zakodowane_tab[firstError]; //ZAMIANA BLEDNYCH BITOW NA POPRAWNE
                                                                      //WARTOSCI
            zakodowane_tab[secondError] = !zakodowane_tab[secondError]; //ZAMIANA BLEDNYCH BITOW NA
                                                                        //POPRAWNE WARTOSCI
            i = (BYTE * 2);
            break;
         }
      }
   }
```

```
        if (blad1 == 1)                         //PRZYPADEK DLA 1 BLEDU
        {
           for (int i = 0; i < (BYTE * 2); i++)
           {
              for (int j = 0; j < BYTE; j++)
              {
                 if (H[j][i] != blad_tab[j])   //WYSZUKIWANIE  KOLUMNY W MACIERZY H TAKIEJ SAMEJ JAK WEKTOR BŁĘDU
                 {
                    break;
                 }
                 if (j == 7)
                 {
                    zakodowane_tab[i] = !zakodowane_tab[i]; //PODMIANA BLEDNEGO BITU NA POPRAWNA WARTOSC
                    i = (BYTE * 2);
                 }
              }
           }
        }
     }
```

Podsumowanie wnioski.

Udało się wykonać zadaną nam implementację kodu. Kod Hamminga działa oraz umożliwia wykrywanie i poprawę błędu, dla każdego kolejnego błędu wystarczy dodać kolejne 4 bit. Zakodowana wiadomość jest dwa razy większa od wiadomości wejściowej.