Transmisja Danych – Lab 08

Krystian Bartosik 213A, nr 44266

Kod źródłowy:

```
// Krystian Bartosik
// bk44266@zut.edu.pl
//   FEDCBA
#define _USE_MATH_DEFINES
#include <iostream>
#include <string>
#include <fstream>
#include "math.h"
#include <complex>
#include <cstddef>
#include <bitset>

using namespace std;

string S2BS(const char* s, string Endian)
{
    string result = "";

    if (Endian == "BigEndian")
    {
        for (int j = 0; j < strlen(s); j++)
        {
            result = result + bitset<8>(s[j]).to_string();
        }
    }

    if (Endian == "LittleEndian")
    {
        for (int j = 0; j < strlen(s); j++)
        {
            result = bitset<8>(s[j]).to_string() + result;
        }
    }
    return result;
}

string ChangeBit(string S, int Bit)
{
    if (S[Bit] == '0')
    {
        S[Bit] = '1';
    }
    else
    {
        S[Bit] = '0';
    }
    return S;
}

int** G_Multiply(int** D) // Przyjmuje 4bitową tablicę
{
    int G[7][4] = { {1,1,0,1},{1,0,1,1},{1,0,0,0},{0,1,1,1},{0,1,0,0},{0,0,1,0},{0,0,0,1} };
    int** C = new int*[8];
    int BitKontrolny = 0;
```

```
    for (int j = 0; j < 8; j++)
        C[j] = new int[1];

    for (int j = 0; j < 8; j++)
    {
        C[j][0] = 0;
    }

    for (int j = 0; j < 7; j++)
    {
        for (int i = 0; i < 4; i++)
        {
            //cout << C[j][0] << "+(" << D[0][i]
            C[j][0] = C[j][0] + (D[i][0] * G[j][i]);
        }
        C[j][0] = C[j][0] % 2;
        BitKontrolny = BitKontrolny + C[j][0];
    }

    C[7][0] = BitKontrolny % 2;

    /*
    for (int j = 0; j < 8; j++) // Wyswietlenie
    {
        for (int i = 0; i < 1; i++)
        {
            cout << C[j][i] << " ";
        }
        cout << endl;
    }
    */

    return C;
}

string Hamming_Nadajnik(string S)
{
    cout << "[NAD] Dane do wyslania:  " << S << endl;
    int** C = new int* [4];

    for (int j = 0; j < 4; j++)
    {
        C[j] = new int[1];
    }

    for (int j = 0; j < 4; j++)
    {
        C[j][0] = (int)S[j] - '0'; // Konwersja na liczbę
    }

    int ** Result = G_Multiply(C);
    string X="00000000";

    for (int j = 0; j < 8; j++)
    {
        X[j] = Result[j][0]+48; // Konwersja na znak
    }
    cout << "[NAD] Nadana wiadomosc:  " << X << endl;
    return X;
}
```

```
string Hamming_Odbiornik(string S)
{
    cout << endl;
    cout << "[ODB] Odebrana wiadomosc:" << S << endl;
    int H[3][7] = { {0,0,0,1,1,1,1},{0,1,1,0,0,1,1},{1,0,1,0,1,0,1} };
    int** C = new int* [8];
    int** Answer = new int* [3];
    int BitParzystosci = 0;

    for (int j = 0; j < 8; j++)
    {
        C[j] = new int[1];
        C[j][0] = (int)S[j] - '0';
        BitParzystosci = BitParzystosci + C[j][0];
    }

    BitParzystosci = BitParzystosci % 2;
    cout << "[ODB] Bit parzystosci:   " << BitParzystosci << endl;

    for (int j = 0; j < 3; j++)
    {
        Answer[j] = new int[1];
        Answer[j][0] = 0;
    }

    cout << "[ODB] Odkodowana macierz:";

    for (int j = 0; j < 3; j++)
    {
        for (int i = 0; i < 7; i++)
        {
            Answer[j][0] = Answer[j][0] + (C[i][0] * H[j][i]);
        }
        Answer[j][0] = Answer[j][0] % 2;
        cout << Answer[j][0];
    }
    cout << endl;

    bool Poprawnosc;
    for (int j = 0; j < 3; j++)
    {
        if (Answer[j][0] == 0)
        {
            Poprawnosc = true;
            continue;
        }
        else
        {
            Poprawnosc = false;
            break;
        }
    }

    if ((Poprawnosc == true) && (BitParzystosci == 0))
    {
        cout << "[ODB] Dane poprawne!" << endl;
        string X = "0000";

        X[0] = C[2][0] + 48;
        X[1] = C[4][0] + 48;
        X[2] = C[5][0] + 48;
        X[3] = C[6][0] + 48;

        cout << "[ODB] Odebrane dane:     " << X << endl << endl;
```

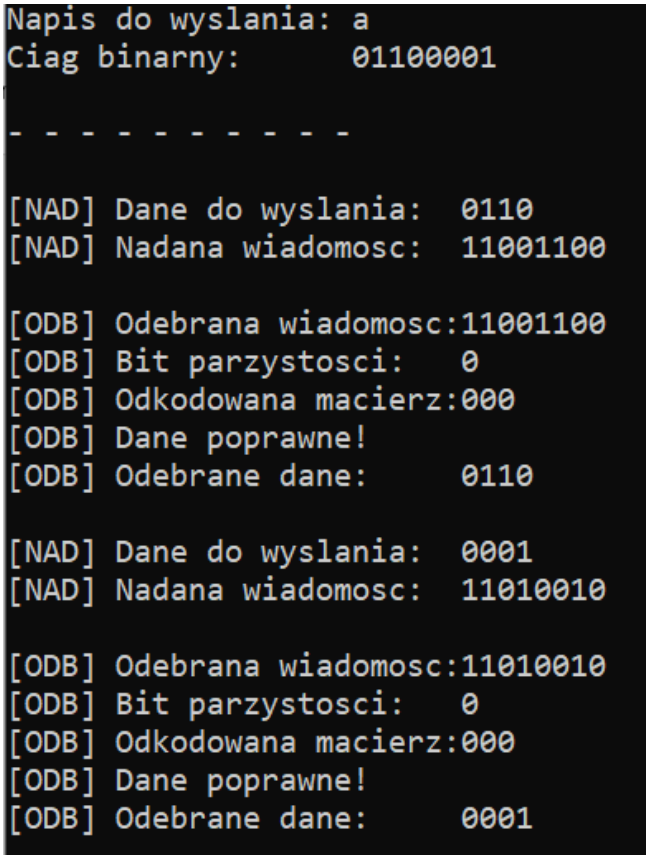
```
        return X;
    }

    if ((Poprawnosc == false) && (BitParzystosci == 1))
    {
        cout << "[ODB] Pojedynczy blad!" << endl;
        int Pozycja = 0;
        for (int j = 0; j < 3; j++)
        {
            Pozycja = Pozycja + (Answer[2-j][0] * pow(2, j));
        }

        cout << "[ODB] Blad na pozycji:   " << Pozycja-1 << endl;
        if (C[Pozycja - 1][0] == 1)
            C[Pozycja - 1][0] = 0;
        else
            C[Pozycja - 1][0] = 1;

        cout << "[ODB] Blad naprawiony!" << endl;

        string X = "0000";

        X[0] = C[2][0] + 48;
        X[1] = C[4][0] + 48;
        X[2] = C[5][0] + 48;
        X[3] = C[6][0] + 48;

        cout << "[ODB] Odebrane dane:     " << X << endl << endl;
        return X;
    }

    if ((Poprawnosc == false) && (BitParzystosci == 0))
    {
        cout << "[ODB] Podwojny blad!" << endl;
        cout << "[ODB] Pakiet odrzucony!" << endl << endl;
        return "Pakiet uszkodzony";
    }
}

int main()
{
    const char * Napis = "a";
    string S = S2BS(Napis, "BigEndian");

    cout << "Napis do wyslania: " << Napis << endl;
    cout << "Ciag binarny:      " << S << endl << endl << "- - - - - - - - - -" << endl << endl;


    int Poz = 0;
    for (int j = 0; j < S.length(); j++)
    {
        if (j % 4 == 0)
        {
            string SS = S.substr(Poz, 4);
            Poz = Poz + 4;
            SS = Hamming_Nadajnik(SS);
            SS = ChangeBit(SS, 5);
            SS = ChangeBit(SS, 3);
            SS = Hamming_Odbiornik(SS);
        }
    }
}
```

Opis kodu:

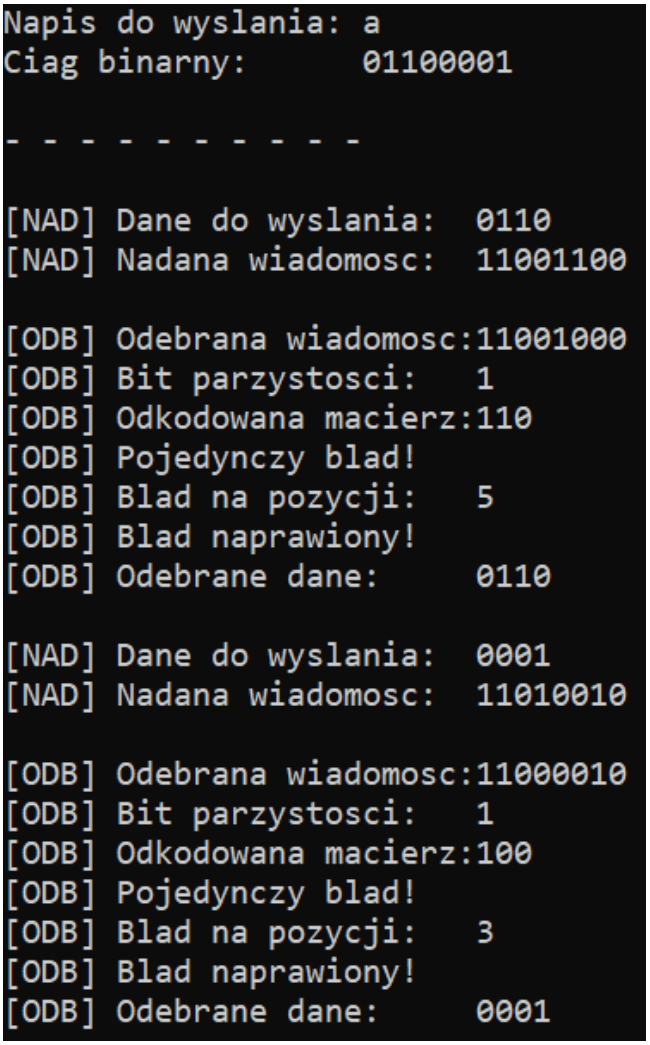
- Funkcja kodująca
- Funkcja odbierająca
- Negacja wskazanego bitu

Poniżej zrzuty ekranu z konsoli. Zostały zaprezentowane 3 przypadki:

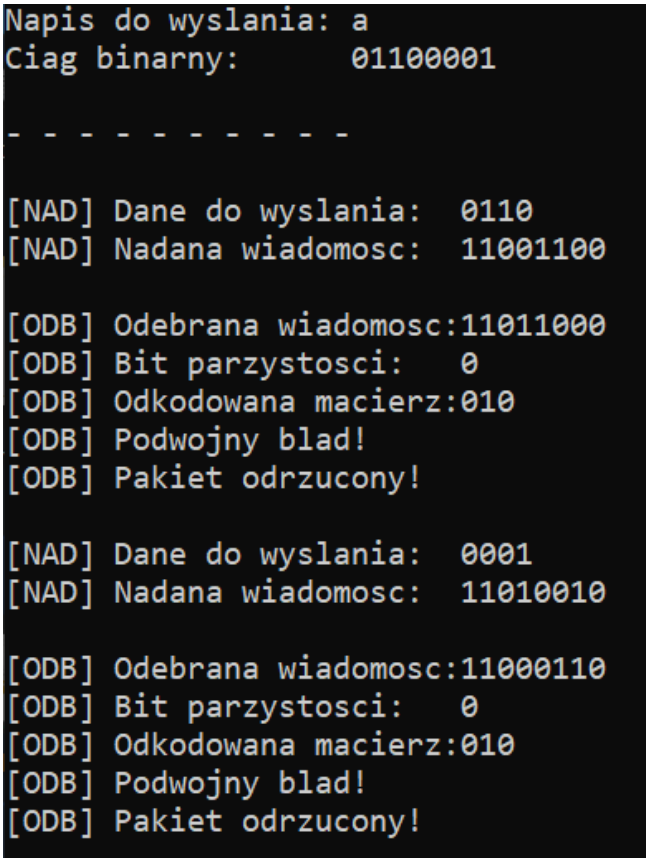
- Wysłanie poprawnych danych
- Wysłanie danych ze zmienionym pojedynczym bitem
- Wysłanie danych ze zmienionymi dwoma bitami



Rysunek 1
Wysłanie poprawnych danych



Rysunek 2
Wysłanie danych ze zmienioną pozycją 5



Rysunek 3
Wysłanie danych ze zmienioną pozycją 3 i 5