- 1. Treść zadania. Napisać program dopisujący do pliku CRC32 oraz sprawdzający integralność danych w pliku z dopisanym CRC.
- 2. Realizacja
- 2.1Metoda realizacji

Program oblicza wartość sumy kontrolnej którą wylicza algorytm CRC z wykorzystaniem funkcji XOR dla bitow zapisanych w pliku tekstowym. Na początku program wczytuje od użytkownika nazwe pliku który chcemy zaszyfrować oraz wielomian CRC (wybralem 0x82608EDB). Po wyślwietleniu wszystkich kroków operacji xor, program wysyła do wybranego przez nas pliku. Program zawiera też metode kontrolną w której sprawdzamy czy zakodowana wiadomość dzieli się bez reszty z wielomianem CRC.

Treść zadania jest zrealizowana w jęzku C++ z użyciem operacji na plikach, operacjach na string oraz pętli.

- 2.2 Ograniczenia metody:
- 2.2.1 Plik wejsciowy musi być liczbą binarną, nie mam zaimplementowanej w kodzie metody zamiany tekstu na ascii a potem na bity. Jeśli chcemy zaszyfrować tekst musimy ją zamienić przed uruchomieniem programu.
- 2.2.2 Chcialem pokazac jak działa crc w kazdym kroku, jednak dla wiekszych kodow nie ma to sensu, dlatego można usunac printf w petlii XOR
- 2.3 Realizacja:

**OBLICZANIE CRC32** 

Main()

{

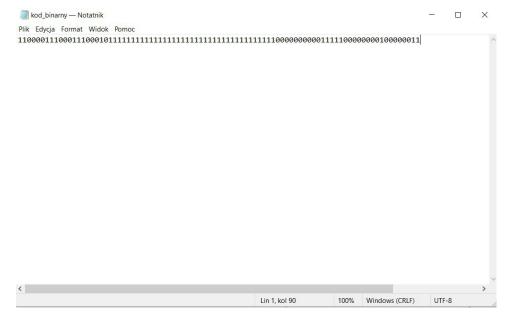
- 2.3.1 Wczytuje plik
- 2.3.2 Dane z pliku zapisuje w stringu

```
2.3.3 Nadaje 32 bitowy wielomian CRC
2.3.4 Dodaje do dlugosci stringa wczytanego 32 zera na koncu
XOR(wczytany string, wybrany wielomian CRC)
{
2.3.5 petla o dlugosci (przemieszczajaca się do 32 miejsca )
2.3.6 xor
}
2.3.7 zapisuje zakodowany tekst do nowego pliku
}
SPRAWDZANIE CRC32
2.3.8 wczytuje plik z kodem źródłowym +CRC
2.3.9 przechodzę do operacji XOR z wielomianem startowym
2.3.10 jeśli operacja XOR dzieli się bez reszty nie ma błędu w
      komunikacji.
Kod źródłowy:
#include<iostream>
#include <fstream>
using namespace std;
string xorfun( string encoded , string crc)
      //Bitowa operacja XOR
{
int crclen = crc.length();
for ( int i = 0 ; i <= (encoded.length() - crclen) ; )</pre>
                                                              // petla
wykonujaca operacje xor
```

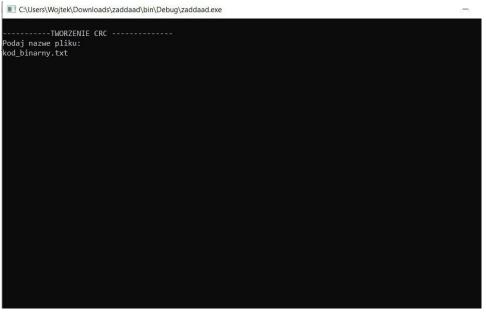
```
{
                         // " 0 xor 0 = 0" " 1 xor 1 = 0 "
                                                                     // " 0 xor 1
for( int j=0 ; j < crclen ; j++)
= 1 " " 1 xor 0 = 1"
{
encoded[i+j] = encoded[i+j] == crc[j] ? '0' : '1' ;
                                                               //jesli bit z pliku i
bit wielomianu jest taki sam to zastap je zerem
      cout<<encded<<endl;</pre>
}
for( ; i< encoded.length() && encoded[i] != '1' ; i++) ;</pre>
}
return encoded;
}
int main()
{
string data , crc , encoded = "";
string myText;
cout<<endl<<"-----"<<endl;
cout<<"Podaj nazwe pliku: "<<endl;</pre>
cin>>data;
  ifstream MyReadFile(data);
  while (getline (MyReadFile, myText)) {
  cout << myText<<endl; }</pre>
  MyReadFile.close();
```

```
crc="10000010011000001000111011011011";
cout<<"Wielomian generujacy: "<<crc<<endl;</pre>
encoded += myText;
int datalen = myText.length();
int crclen = crc.length();
for(int i=1; i <= (crclen - 1); i++)
encoded += '0';
encoded = xorfun(encoded , crc);
cout<<"Cykliczny kod nadmiarowy to: ";</pre>
cout<<encoded.substr(encoded.length() - (crclen-1) )<<endl;</pre>
cout<<myText + encoded.substr(encoded.length() - crclen + 1)<<endl;</pre>
      string data2;
      cout<<"Podaj nazwe pliku do zapisania: "<<endl;</pre>
      cin>>data2;
  ofstream MyFile(data2);
  MyFile <<(myText + encoded.substr(encoded.length() - crclen + 1));
  MyFile.close();
```

```
cout<<endl<<"-----"<<endl;
  cout<<"Podaj nazwe pliku z zaszyfrowana wiadomoscia: "<<endl;</pre>
  string msg, data1;
  cin>>data1;
  ifstream MyReadFile1(data1);
  getline (MyReadFile1, msg);
  MyReadFile1.close();
cout<<"Zakodowana wiadomosc to: "<<endl<<msg<<endl<<endl;</pre>
msg = xorfun( msg , crc);
for( char i : msg.substr(msg.length() - crclen + 1))
if( i != '0' )
{
cout<<"Blad w komunikacji "<<endl;</pre>
return 0;
}
cout<<"Brak bledu !"<<endl;
return 0;
}
2.3 Uruchomiony program:
2.3.1 Plik z liczba binarną:
```



# 2.3.2 Wczytywanie pliku



# 2.3.4Wyliczony 31 bitowy kod CRC

# a) z każdą kolejną operacją



### b) bez wypisywania operacji

WybierzC:\Users\Wojtek\Downloads\zaddaad\bin\Debug\zaddaad.exe ---TWORZENIE CRC --

2.3.5

Podaj	nazwe	pliku	do	zapisania:
kod_binarny+CRC.txt				

-----SPRAWDZENIE------

2.3.6 Wczytujemy plik w którym zapisaliśmy kod binarny z obliczonym CRC

```
odaj nazwe pliku z zaszyfrowana wiadomoscia:
od binarny-KR.txt
okodowana wiadomosc
00000111110000
00011111000
                                       90001000000111101111110100101001010010100010
00011111000
                                           901111100
                                           001111011111101001010010100101000010
0011111000
0011111000
                                 001111100
001111100
                                        99199
                                           00011110111111010010100101001010010100010
01111100
901111100
                                        90100
                                           00011110111111010010100101001010010
                                 0011111000
0011111000
                                           00011110111111010010100101001010010
                                 001111100
                                        00100
9000111101111111010010100101001010010
900011110111111101001010010100101000010
                                 9911111996
001111011111110100101001010010100010
                                 901111100
0011111000
                                        99199
                                 0011111000
0011111000
001111100
                                        00100
                                           000111101111110100101001010010100010
901111100
901111100
                                           900111101111111010010100101001010010
9001111011111110100101001010010100010
901111100
                                           00111101111111010010100101001010010
0011111000
0011111000
0011111000
                                           90011110111111101001010001010010100010
9001111011111110100101001010010100010
9001111011111110100101001010010100010
9911111996
                                        99199
                                           00011110111111010010100101001010010
 901111100
901111100
                                           901111100
                                        0100
                                           00111101111110100101001010010100010
 9001111011111110100101000101000101
90011110111111101001010001010010100010
9001111011111110100101001010010100010
                                 9011111006
9011111006
                                 901111100
 9011111000
                                        99199
                                           9001111011111110100101001010010100010
  00111101111110100101001010010100010
  0011111000
                                       001000000111101111110100101001010010100010
  0011111000
0011111000
```

#### 2.3.7 Poprawny kod:

2.3.7 Podmieniony kod (bez wypisywania wszytkich operacji xor):