

SPRAWOZDANIE Z 2 ZADANIA LABORATORYJNEGO Z PRZEDMIOTU TEORIA INFORMACJI I KODOWANIA.

Treść zadania:

Napisz program dopisujący do pliku CRC32 oraz sprawdzający integralność danych w pliku z dopisanym CRC.

Analiza zadania

Kody cykliczne wykorzystywane są do weryfikacji integralności danych. Mianem CRC określa się funkcję która dla zadanego ciągu danych binarnych oblicza ciąg kontrolny o ustalonej długości. Ciąg kontrolny jest dołączany do ciągu danych. Ciąg kontrolny nazywamy wartością CRC. Wartość CRC wykorzystujemy do sprawdzenia, czy dane nie zostały naruszone. Sprawdzenie polega na obliczeniu ciągu kontrolnego dla ciągu odebranego. Uzyskanie niezerowej wartości CRC świadczy o naruszeniu integralności danych. W ten sposób można wykryć przypadki błędów czy przeróbek danych dokonanych podczas transmisji lub przechowywania.

Założenia:

Wielomian generujący standard CRC-32-IEEE 802.3 (Ethernet), RCR-32 CCITT ma postać

$$g(x)=x^{32}+x^{26}+x^{23}+x^{22}+x^{16}+x^{12}+x^{11}+x^{10}+x^8+x^7+x^5+x^4+x^2+x+1$$

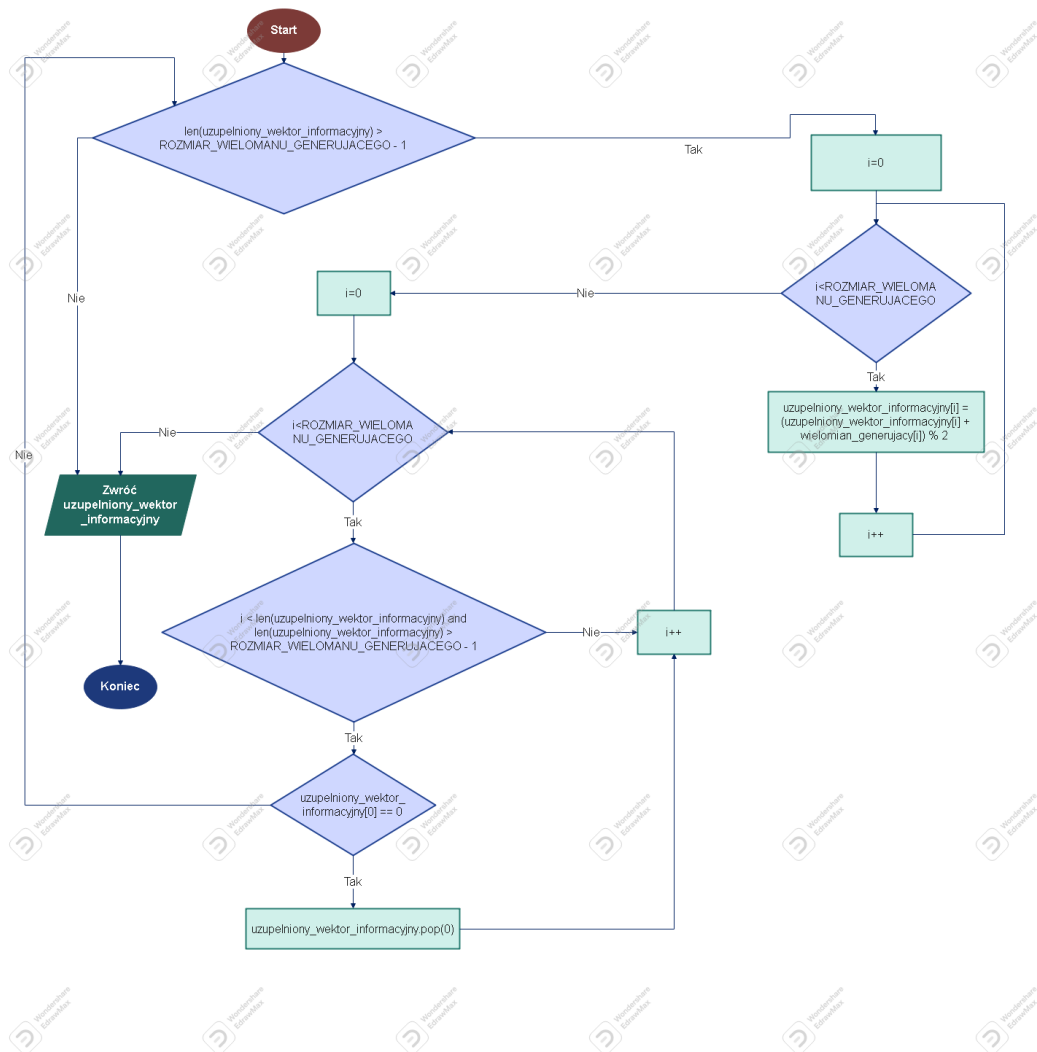
Wielomian generujący zapisywany jest w postaci ciągu binarnych współczynników wielomianu.

Kod CRC wyznaczany jest jako reszta z dzielenia uzupełnionego wielomianu informacyjnego przez wielomian generujący.

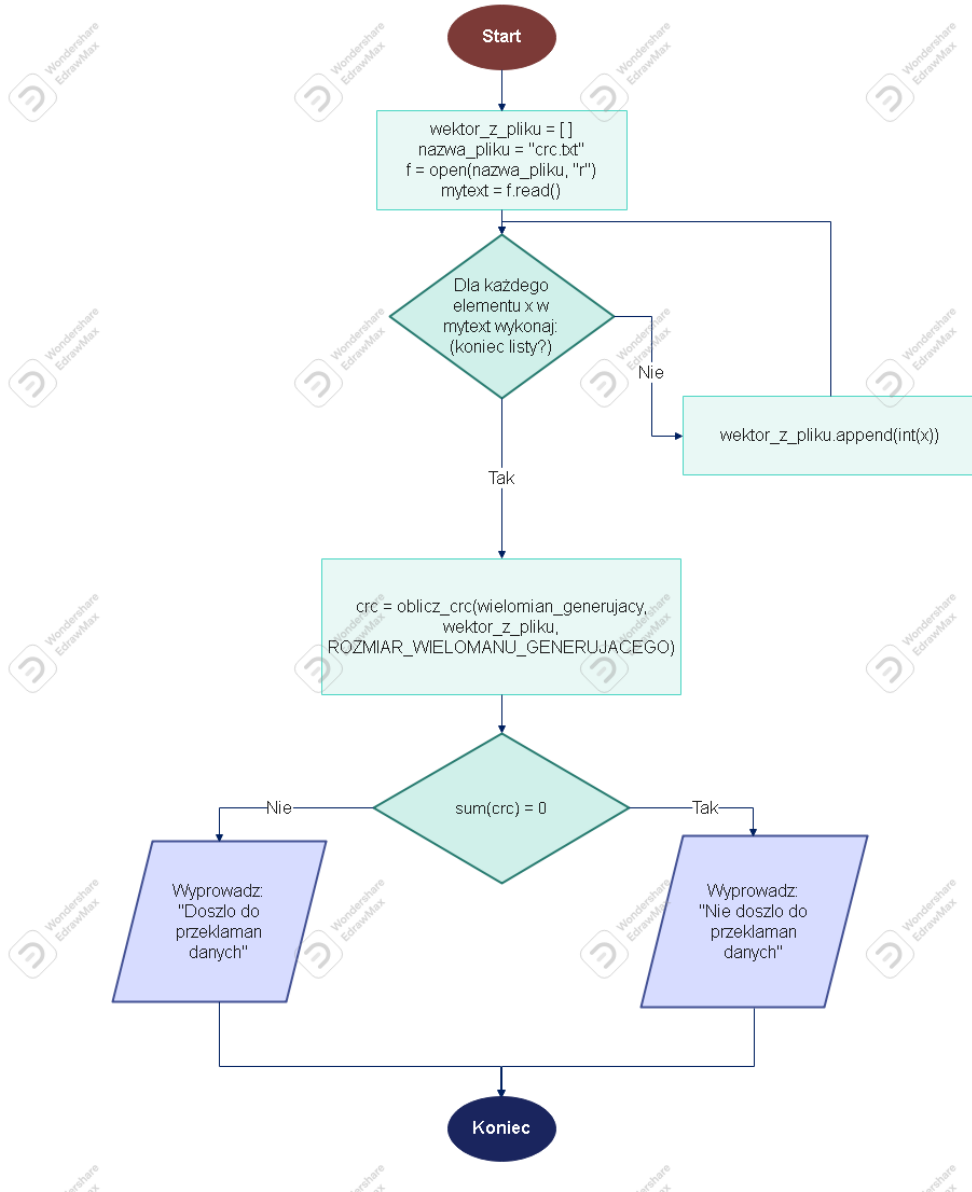
Algorytm

Schematy blokowe wykonałem w programie WondershareEdrawMax (darmowa licencja próbna, znak wodny).

Schemat blokowy funkcji crc



Schemat blokowy sprawdzania integralności



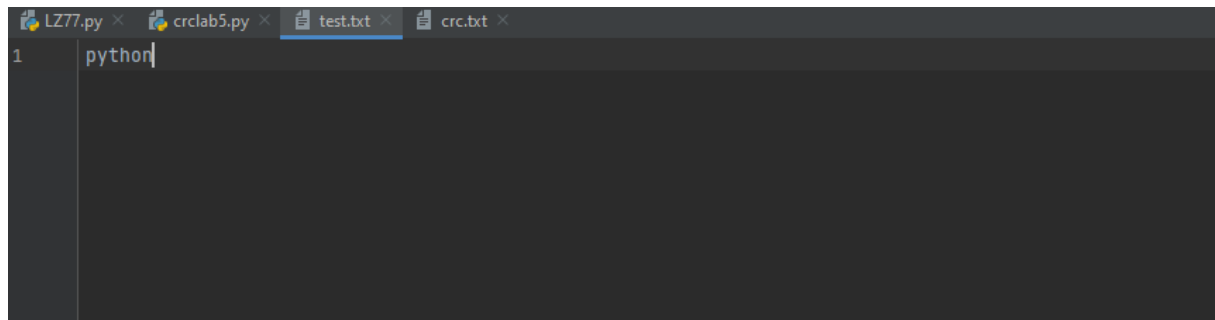
Implementacja

Początkowo algorytm chciałem implementować w języku c++, jednak ze względu na swoje ograniczenia związane z tym językiem, postanowiłem napisać kod w języku python. Korzystałem ze środowiska PyCharm community edition.

Użytkownik określa nazwę pliku dla który zostanie przekonwertowany na ciąg binarny, a także nazwę pliku do którego zostaną zapisane dane w postaci binarnej, a także nazwę pliku do którego zostanie dopisane crc i nazwę pliku dla którego zostanie sprawdzona integralność danych. Po sprawdzeniu integralności danych w terminalu pojawia się odpowiedni komunikat, mówiący o tym czy doszło do przekłamań danych, czy też nie.

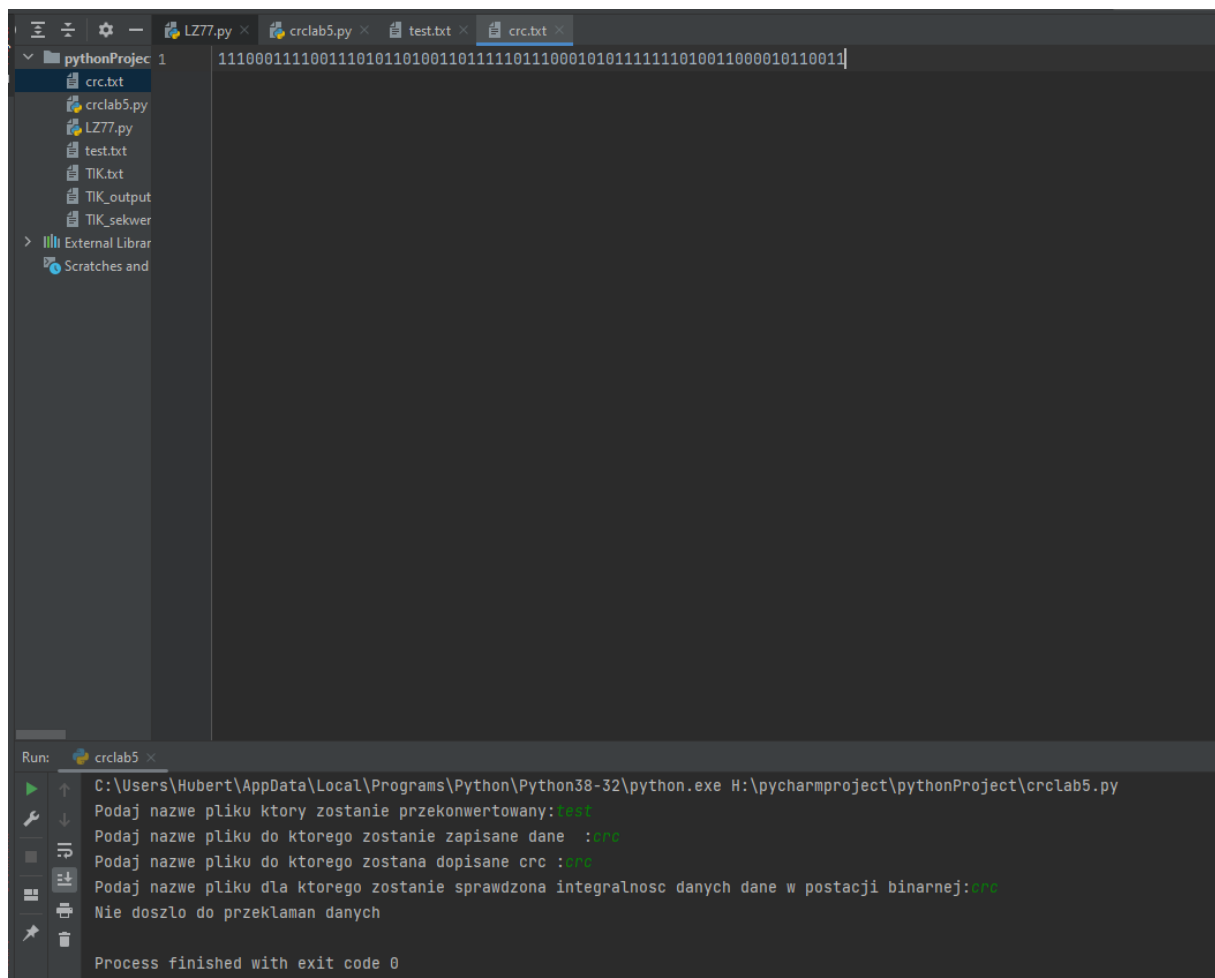
Testowanie

Zawartość pliku test.txt



A screenshot of a code editor with four tabs: LZ77.py, crclab5.py, test.txt, and crc.txt. The test.txt tab is active, showing a single line of code: `python` at line 1.

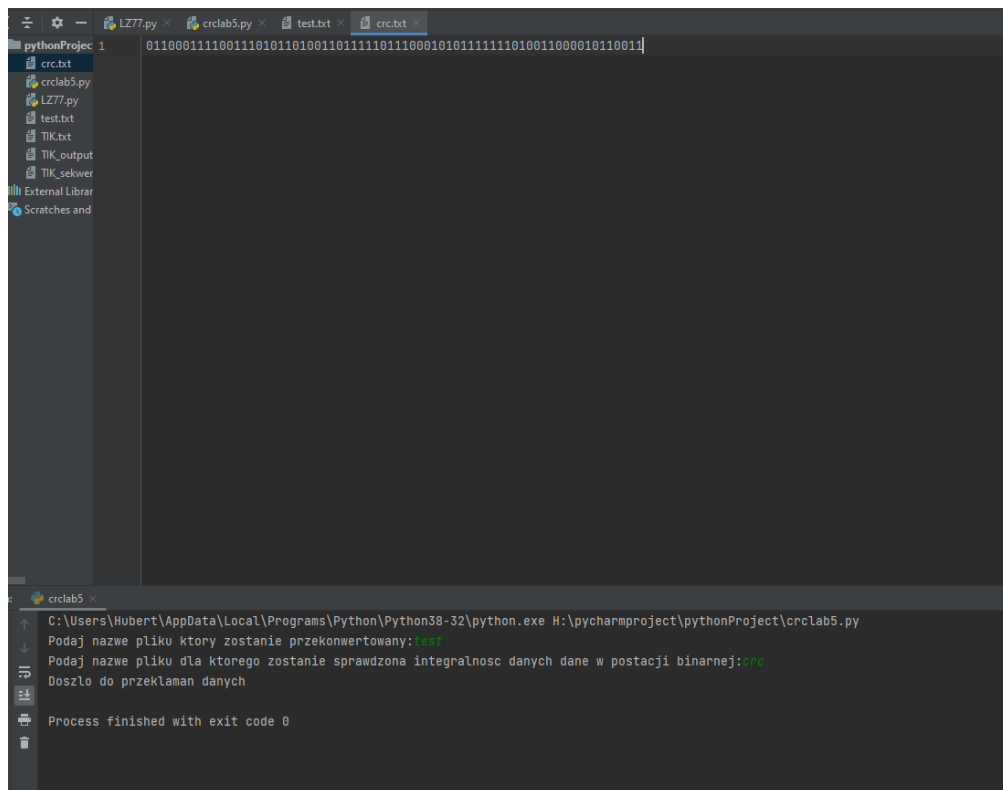
Zawartość pliku crc.txt po dopisaniu crc



A screenshot of a code editor with the same four tabs. The crc.txt tab is active, showing a long binary string: `1110001111001110101101001101111101110001010111111010011000010110011`. The Run console at the bottom shows the output of the script crclab5.py, which prompts for file names and prints the CRC value.

```
Run: crclab5
C:\Users\Hubert\AppData\Local\Programs\Python\Python38-32\python.exe H:\pycharmproject\pythonProject\crclab5.py
Podaj nazwe pliku ktory zostanie przekonwertowany: test
Podaj nazwe pliku do ktorego zostanie zapisane dane : crc
Podaj nazwe pliku do ktorego zostana dopisane crc : crc
Podaj nazwe pliku dla ktorego zostanie sprawdzona integralnosc danych dane w postaci binarnej: crc
Nie doszlo do przeklamania danych
Process finished with exit code 0
```

Po zmianie pierwszego bita w pliku crc.txt

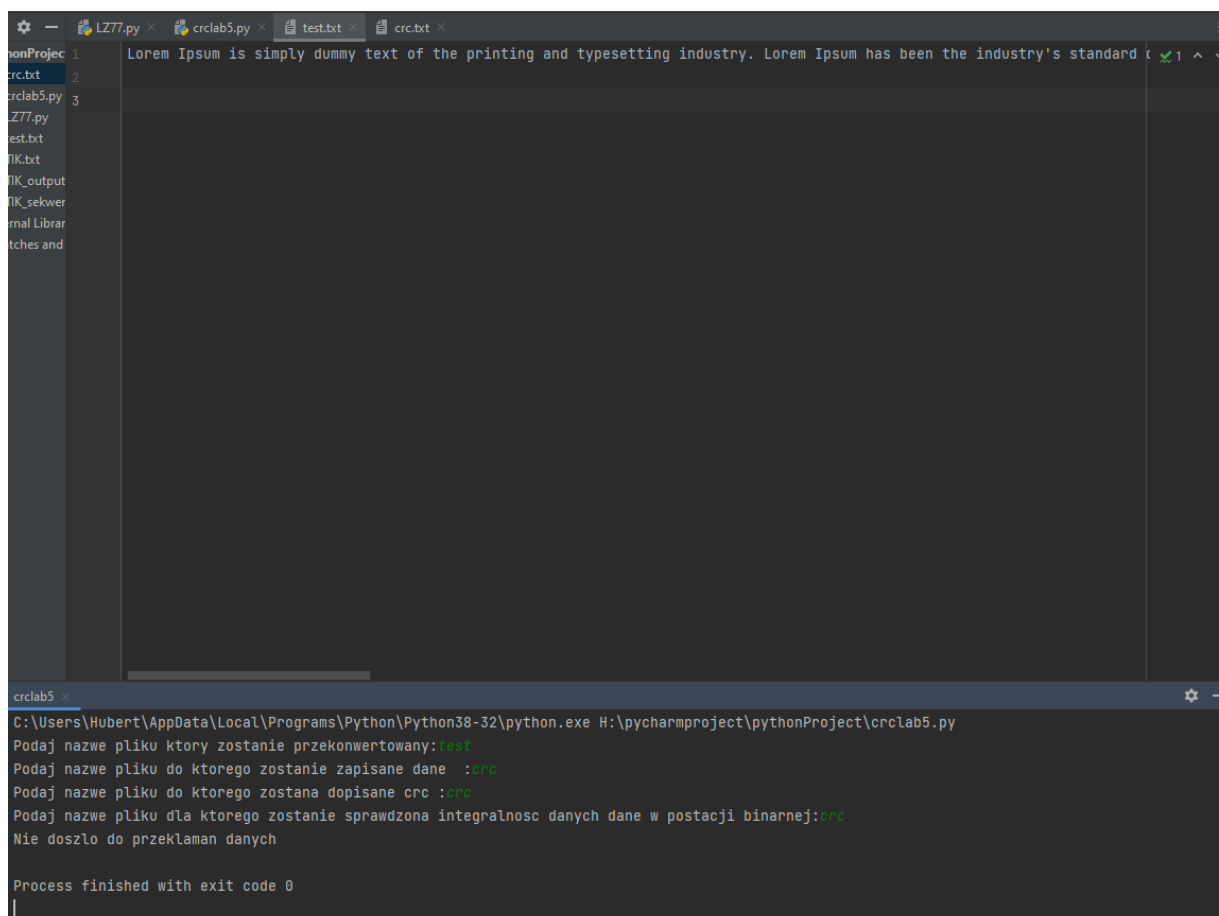


The screenshot shows the PyCharm IDE interface. The file editor at the top displays the contents of `crc.txt`, which is a long string of binary digits (0s and 1s). The file explorer on the left shows the project structure, including `pythonProject` and its subfolders. Below the editor, the terminal window shows the output of a Python script named `crclab5.py`. The script prompts the user for a file name to be converted, which is `test`, and a file name for the CRC output, which is `orc`. It then reports that the data conversion was successful and that the process finished with exit code 0.

```
01100011110011101011010011011111011100010101111111010011000010110011
```

```
C:\Users\Hubert\AppData\Local\Programs\Python\Python38-32\python.exe H:\pycharmproject\pythonProject\crclab5.py
Podaj nazwe pliku ktory zostanie przekonwertowany: test
Podaj nazwe pliku dla ktorego zostanie sprawdzona integralnosc danych dane w postaci binarnej: orc
Doszlo do przeklaman danych
Process finished with exit code 0
```

2 przypadek dla większego pliku tekstowego, zawartość pliku test.txt

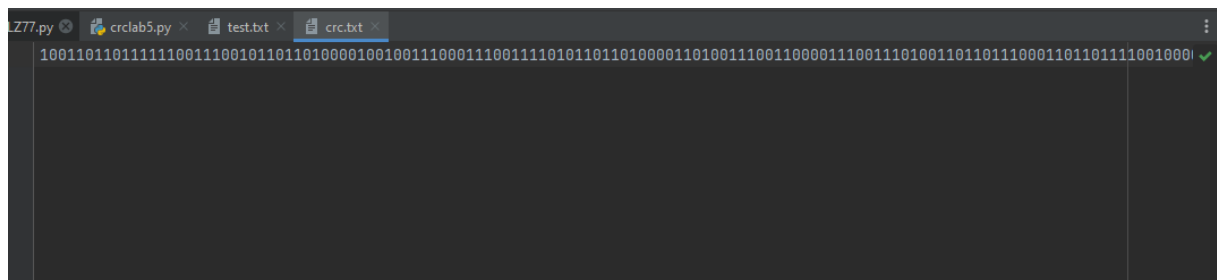


The screenshot shows the PyCharm IDE interface. The file editor at the top displays the contents of `test.txt`, which is a paragraph of Lorem Ipsum text. The file explorer on the left shows the project structure, including `pythonProject` and its subfolders. Below the editor, the terminal window shows the output of a Python script named `crclab5.py`. The script prompts the user for a file name to be converted, which is `test`, and a file name for the CRC output, which is `orc`. It then reports that the data conversion was successful and that the process finished with exit code 0.

```
Lorem Ipsum is simply dummy text of the printing and typesetting industry. Lorem Ipsum has been the industry's standard
```

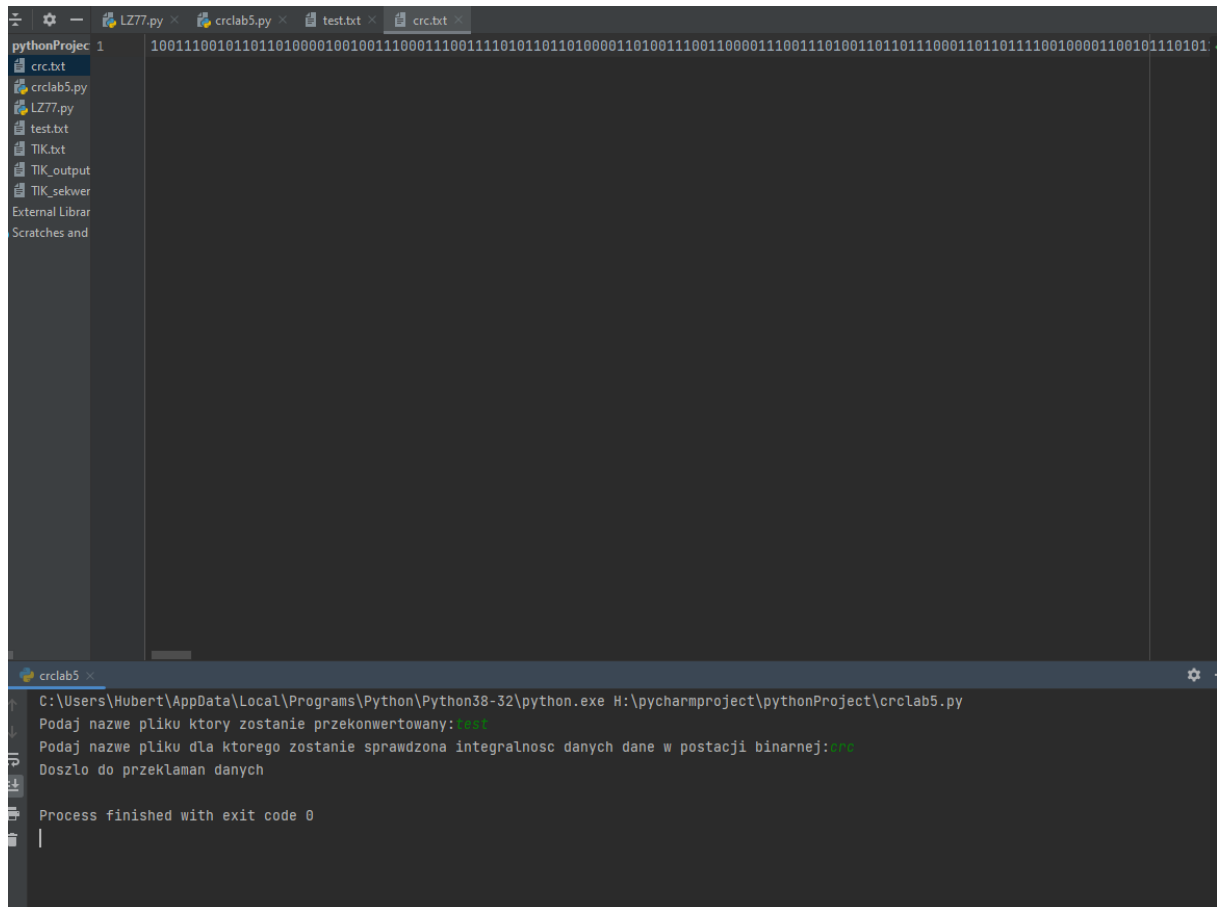
```
C:\Users\Hubert\AppData\Local\Programs\Python\Python38-32\python.exe H:\pycharmproject\pythonProject\crclab5.py
Podaj nazwe pliku ktory zostanie przekonwertowany: test
Podaj nazwe pliku do ktorego zostanie zapisane dane :orc
Podaj nazwe pliku do ktorego zostana dopisane crc :orc
Podaj nazwe pliku dla ktorego zostanie sprawdzona integralnosc danych dane w postaci binarnej:orc
Nie doszlo do przeklaman danych
Process finished with exit code 0
```

Część danych w pliku crc.txt



```
10011011011111110011100101101101000010010011100011100111101011011010000110100111001100001110011101001101101110001101101110010001
```

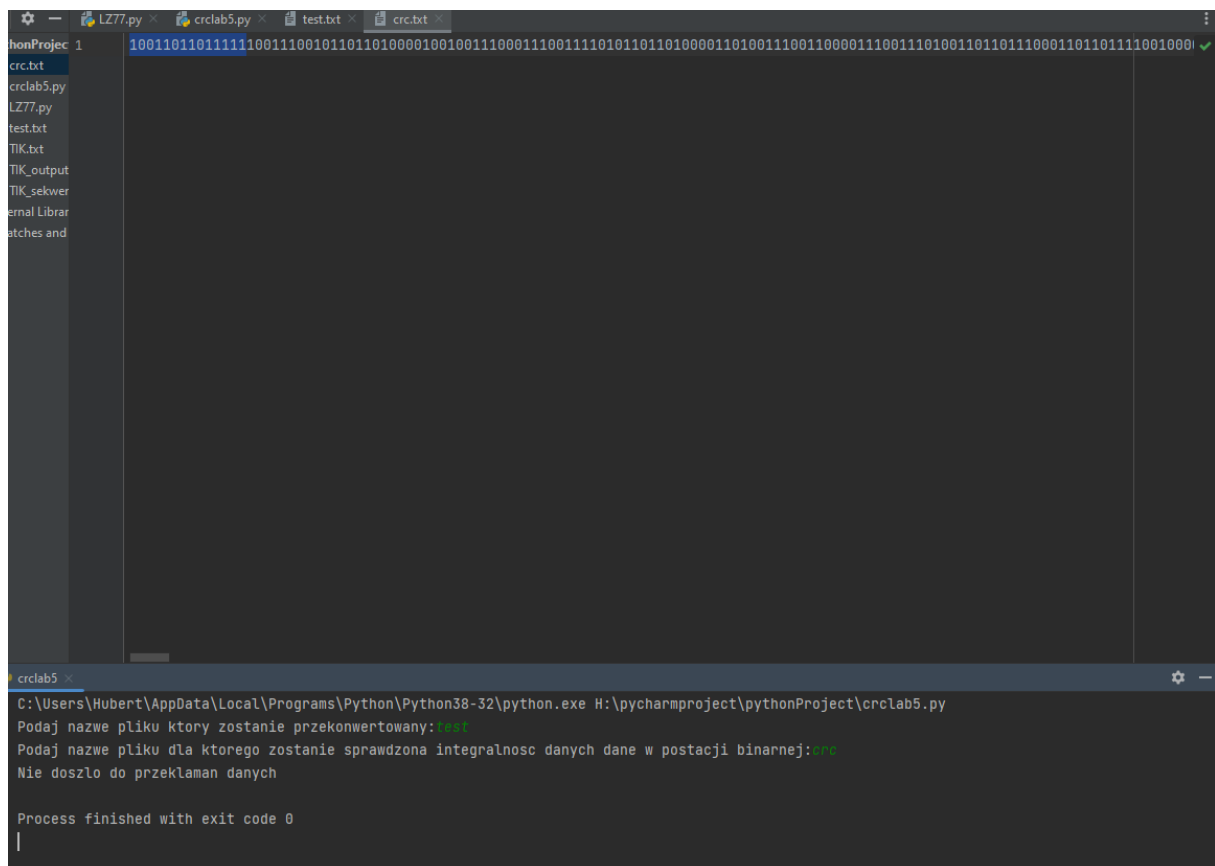
Sprawdzenie integralności po usunięciu pierwszych 14 bitów plik crc.txt



```
1001110010110110100001001001110001110011110101101101000011010011100110000111001110100110110111000110110111100100001100101110101
```

```
C:\Users\Hubert\AppData\Local\Programs\Python\Python38-32\python.exe H:\pycharmproject\pythonProject\crcclab5.py
Podaj nazwe pliku ktory zostanie przekonwertowany: test
Podaj nazwe pliku dla ktorego zostanie sprawdzona integralnosc danych dane w postaci binarnej: crc
Doszlo do przeklamania danych
Process finished with exit code 0
```

Po przywróceniu bitów plik crc.txt



The image shows a PyCharm IDE interface. The top toolbar includes icons for settings, a file explorer, and tabs for 'LZ77.py', 'crclab5.py', 'test.txt', and 'crc.txt'. The left sidebar shows a project view with files like 'crc.txt', 'crclab5.py', 'LZ77.py', 'test.txt', 'TIK.txt', 'TIK_output', 'TIK_sekwer', 'ernal Librar', and 'atches and'. The main editor window displays a single line of binary data: '1001101101111111001110010110110100001001001110001110011110101101101000011010011100110000111001110100110110111000110110111001000'. The bottom terminal window shows the command 'C:\Users\Hubert\AppData\Local\Programs\Python\Python38-32\python.exe H:\pycharmproject\pythonProject\crclab5.py' and its output: 'Podaj nazwe pliku ktory zostanie przekonwertowany: test', 'Podaj nazwe pliku dla ktorego zostanie sprawdzona integralnosc danych dane w postaci binarnej: crc', 'Nie doszlo do przeklamania danych', and 'Process finished with exit code 0'.

```
1001101101111111001110010110110100001001001110001110011110101101101000011010011100110000111001110100110110111000110110111001000
```

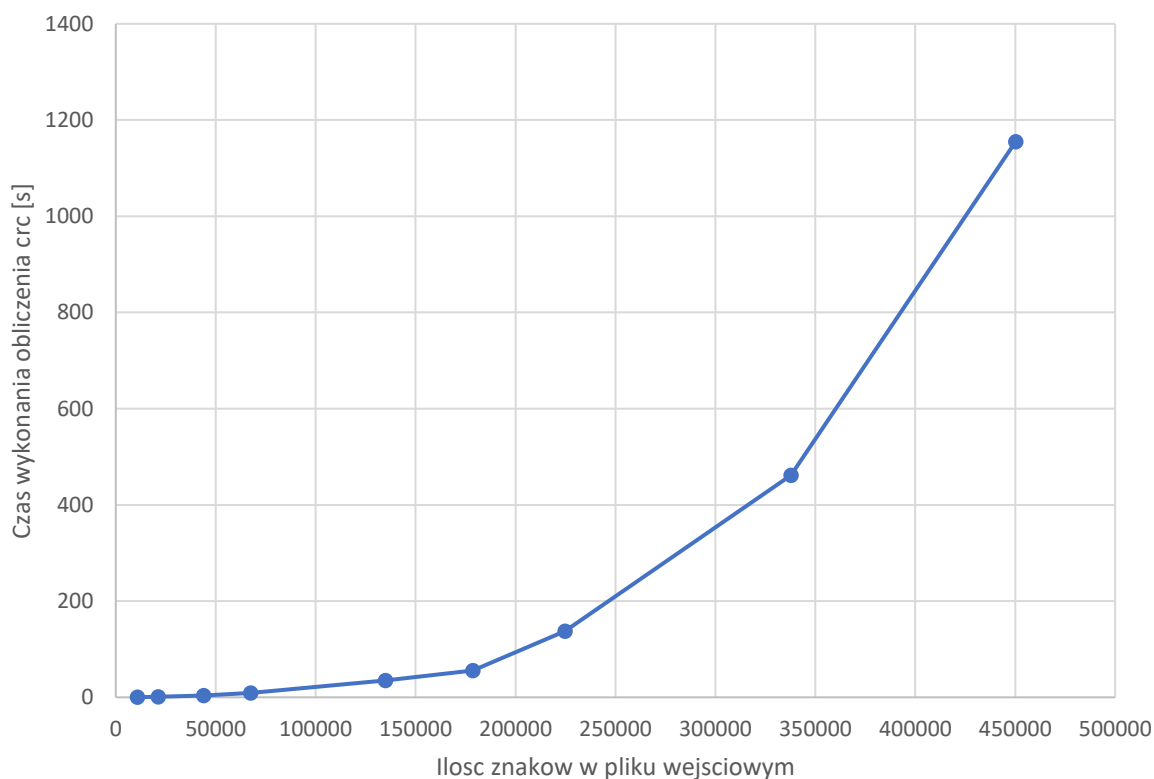
```
C:\Users\Hubert\AppData\Local\Programs\Python\Python38-32\python.exe H:\pycharmproject\pythonProject\crclab5.py
Podaj nazwe pliku ktory zostanie przekonwertowany: test
Podaj nazwe pliku dla ktorego zostanie sprawdzona integralnosc danych dane w postaci binarnej: crc
Nie doszlo do przeklamania danych

Process finished with exit code 0
```


Testowanie czasu wykonania algorytmu

Tabela 1. Zestawienie wyników otrzymanych dla czasu wykonania funkcji crc dla różnej ilości danych w pliku wejściowym	
Ilość znaków w pliku wejściowym	Czas wykonania obliczenia crc [s]
450218	1155
337836	461.7
224749	137.7
178541	56
134842	35
67421	8.8
43888	3.9
21132	1
10711	0.3

Wykres 1. Zestawienie wyników otrzymanych dla wykonania funkcji crc dla różnej ilości danych w pliku wejściowym



Wnioski

Algorytm poprawnie wskazuje przekłamania danych.

Złożoność obliczeniowa funkcji obliczenia crc szacuje na $T(n)=O(n^2)$, wykres zdaje się również potwierdzać to założenie. Ze względu na złożoność kwadratową, algorytm nie byłby odpowiedni do sprzedaży komercyjnej, jednak na potrzeby dydaktyczne spełnia swoje założenia. W celu poprawienia wydajności algorytmu można by zastosować tablicowanie wartości CRC.

Kod programu

```
def zapisz_crc_plik(crc): #dopisz crc do pliku
    tmp = ""
    for x in crc:
        tmp += str(x)
    nazwa_pliku = input("Podaj nazwe pliku do ktorego zostanie dopisane
crc: ")
    nazwa_pliku += ".txt"
    try:
        filetest = open(nazwa_pliku, "a")
    except:
        print("Problem z otwarciem pliku")
        exit()
    filetest.write(tmp)
    filetest.close()

def zapisz_dane_plik(tmp): #przepisz dane w formie binarnej z pliku
wejsciowego do pliku crc.txt
    nazwa_pliku = input("Podaj nazwe pliku do ktorego zostana zapisane
dane: ")
    nazwa_pliku += ".txt"
    try:
        filetest = open(nazwa_pliku, "w")
    except:
        print("Problem z otwarciem pliku")
        exit()
    filetest.write(tmp)
    filetest.close()

def oblicz_crc(wielomian_generujacy,uzupelniony_wektor_informacyjny,
ROZMIAR_WIELOMANU_GENERUJACEGO):
    while (len(uzupelniony_wektor_informacyjny) >
ROZMIAR_WIELOMANU_GENERUJACEGO - 1):
        for i in range(ROZMIAR_WIELOMANU_GENERUJACEGO):
            uzupelniony_wektor_informacyjny[i] =
(uzupelniony_wektor_informacyjny[i] + wielomian_generujacy[i]) % 2
#dodawanie modulo 2 odpowiednich bitow
        for i in range(ROZMIAR_WIELOMANU_GENERUJACEGO):
            if (i < len(uzupelniony_wektor_informacyjny) and
len(uzupelniony_wektor_informacyjny) > ROZMIAR_WIELOMANU_GENERUJACEGO - 1):
                if (uzupelniony_wektor_informacyjny[0] == 0): #usun z listy
zerowe bity zaczynajac od poczatku listy do napotkania 1
                    uzupelniony_wektor_informacyjny.pop(0)
                else:
                    break
        return uzupelniony_wektor_informacyjny #zwroc crc

def sprawdz_integralnosc(wielomian_generujacy,
ROZMIAR_WIELOMANU_GENERUJACEGO):
    wektor_z_pliku = []
```

```

nazwa_pliku = input("Podaj nazwe pliku dla ktorego zostanie sprawdzona
integralnosc danych dane w postaci binarnej: ")
nazwa_pliku += ".txt"
try:
    f = open(nazwa_pliku, "r")
except:
    print("Problem z otwarciem pliku")
    exit()
mytext = f.read()
for mychar in mytext: #czytaj z pliku
    if (mychar not in ['0', '1']):
        print("Plik nie ma odpowiedniego formatu")
        exit()
    wektor_z_pliku.append(int(mychar))
f.close()
crc = oblicz_crc(wielomian_generujacy, wektor_z_pliku,
ROZMIAR_WIELOMANU_GENERUJACEGO)
if (sum(crc) == 0): #sprawdz czy crc wynosi 0
    print("Nie doszlo do przeklamania danych")
else:
    print("Doszlo do przeklamania danych")

#main function
ROZMIAR_WIELOMANU_GENERUJACEGO = 33
wektor_z_pliku = []
nazwa_pliku = input("Podaj nazwe pliku ktory zostanie przekonwertowany: ")
nazwa_pliku += ".txt"
try:
    f = open(nazwa_pliku, "r") # otworz plik z ktorego proces bedzie czytal
znaki
except:
    print("Problem z otwarciem pliku")
    exit()

mytext = f.read() #czytaj z pliku
tmp = ""
for mychar in mytext:
    for mybin in bin(ord(mychar))[2:-1]: # zamien na zapis binarny usun
'0b'
        wektor_z_pliku.append(int(mybin))
        tmp += mybin

crc = []
wielomian_generujacy = []
for i in range(ROZMIAR_WIELOMANU_GENERUJACEGO): #wypelnij zerami liste crc
i wielomian generujacy
    wielomian_generujacy.append(0)
    crc.append(0)
crc.pop() #usun element z crc
#wielomian generujacy CRC-32-IEEE 802.3, RCR-32 CCITT
 $x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$ 
for i in range(ROZMIAR_WIELOMANU_GENERUJACEGO): #wypelnij wielomian
generujacy
    if i in [0, 1, 2, 4, 5, 7, 8, 10, 11, 12, 16, 22, 23, 26, 32]:
        wielomian_generujacy[i] = 1
    else:
        wielomian_generujacy[i] = 0

for x in crc: #uzupelnij wektor o zerowe crc
    wektor_z_pliku.append(0)
zapisz_dane_plik(tmp) #odkomentowanie jesli nie dodalismy crc

```

```
zapisz_crc_plik(oblicz_crc(wielomian_generujacy, wektor_z_pliku,  
ROZMIAR_WIELOMANU_GENERUJACEGO)) #odkomentowanie jesli nie dodalismy crc  
sprawdz_integralnosc(wielomian_generujacy, ROZMIAR_WIELOMANU_GENERUJACEGO)
```