

Dokumentacja Projektu

Języki Skryptowe

Paweł Pietraszko, Gr III F

Opis działania programu

Program wczytuje tekst z pliku wejściowego *Tekst.txt* a następnie szyfruje go i zapisuje w pliku *Szyfr.txt*, a następnie za pomocą algorytmu deszyfrującego odtwarza pierwotną formę i zapisuje w pliku *Odszyfrowane.txt* i tworzy plik *index.html* zawierający dane z pliku *Szyfr.txt* oraz *Deszyfrowane.txt* odpowiednio sformatowaną z pomocą CSS'a.

Założenia i funkcjonalność programu oparta jest na poleceniu do zadania nr 1 - "Szyfrowanie wiadomości" z konkursu *Algorytmion* 2015.

ZADANIE 1 - "SZYFROWANIE WIADOMOŚCI"

Zadanie zaproponował: mgr Krzysztof Jarczewski, III LO im. S. Batorego w Chorzowie

Adam i Janek ustalili między sobą, że każdą wiadomość tekstową, będą szyfrować przy pomocy następującego sposobu:

- Każdą literę i znak interpunkcyjny należy zamienić na odpowiadającą liczbę kodu ASCII (liczba naturalna od 0 do 127).
- Pierwsza, tak powstała liczba, jest zamieniana na system dwójkowy.
- Kolejne liczby, wynikające z kodu ASCII, są zamieniane na system liczbowy, który jest równy powiększonej o dwa reszcie z dzielenia przez osiem poprzedniej liczby.
- Ilość cyfr zamienionej liczby, dla każdego systemu liczbowego, wynika z ilości cyfr zamiany liczby maksymalnej, czyli liczby 127 (dla systemu binarnego jest to siedem cyfr).

Twoim zadaniem jest napisanie programu, który szyfruje i deszyfruje wiadomości.

Szyfrowanie tekstu z pliku *tekst.txt* do pliku *szyfr.txt*

Deszyfrowanie tekstu z pliku *szyfr.txt* do pliku *odszyfrowane.txt*

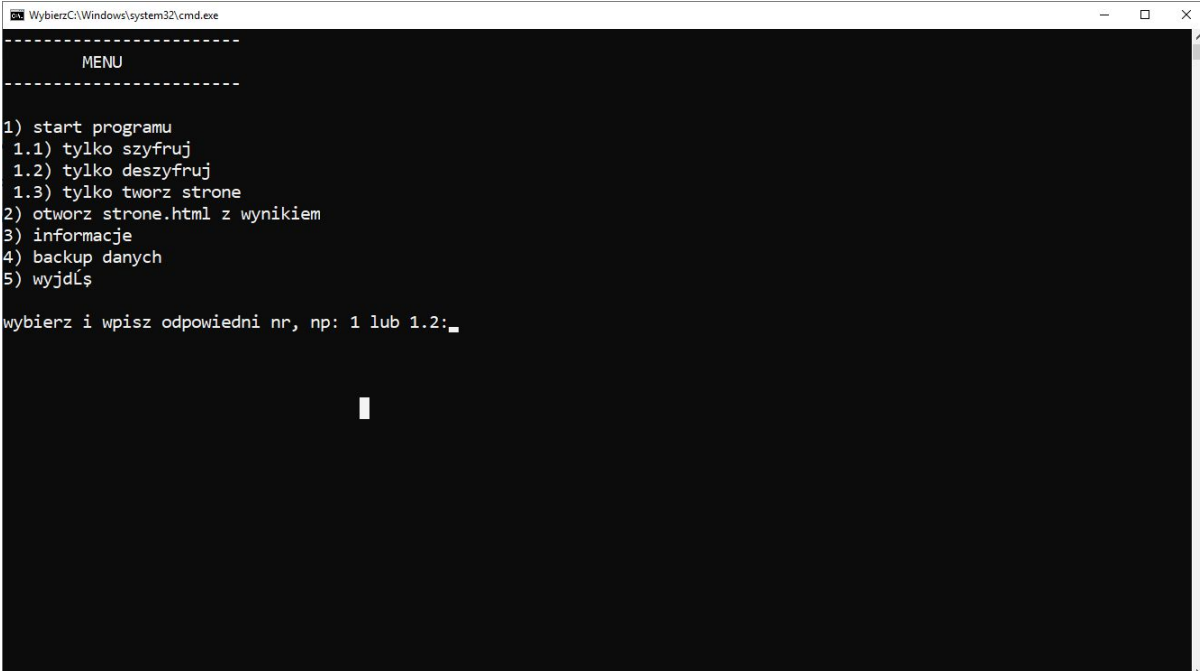
Przykład

tekst	kod ASCII	system liczbowy: 2, $(66 \bmod 8)+2=4$, $(105 \bmod 8)+2=3$
Bit	→ 66 105 116	→ 1000010 1221 11022

Każda linijka podczas szyfrowania i deszyfrowania jest traktowana jako osobna wiadomość. Ma to znaczenie, gdyż każdy pierwszy znak szyfrowany jest w taki sam sposób, a na jego podstawie jego kodu ASCII wyliczany jest sposób szyfrowania kolejnego znaku.

Obsługa programu

Do obsługi funkcjonalności, jaką oferuje program, najlepiej jest wykorzystać proste menu *menu.bat*:



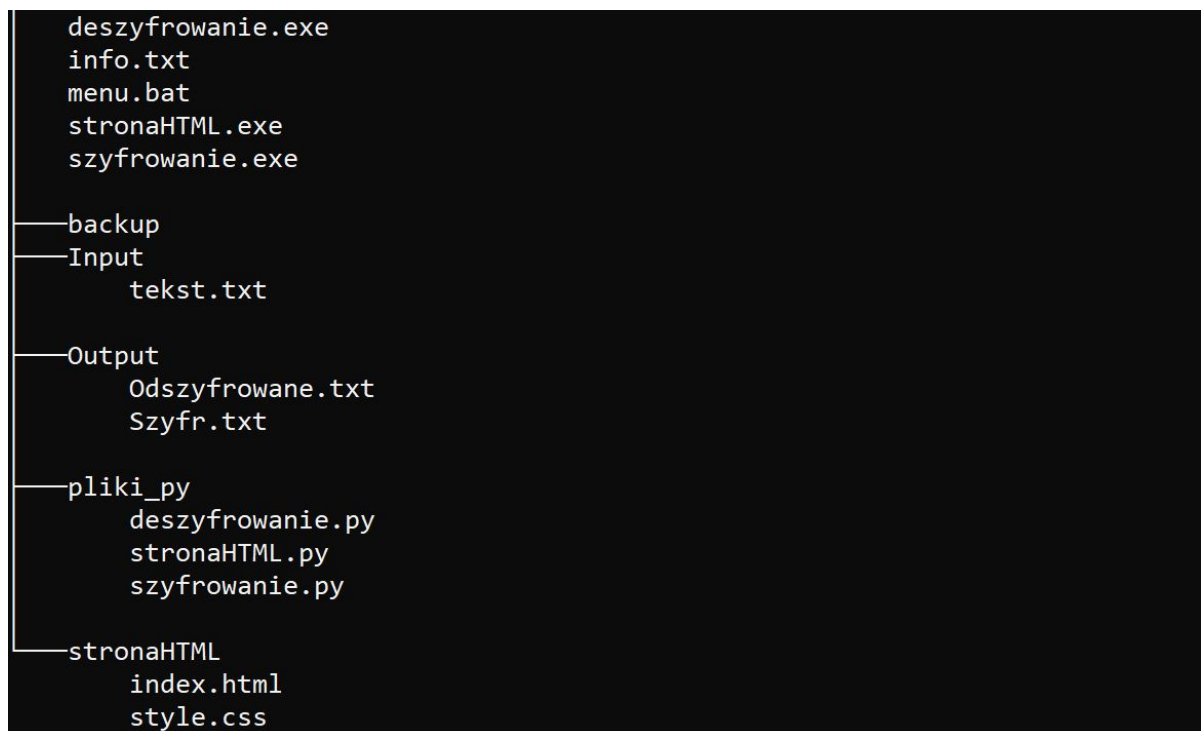
```
WybierzC:\Windows\system32\cmd.exe
-----
MENU
-----
1) start programu
  1.1) tylko szyfruj
  1.2) tylko deszyfruj
  1.3) tylko tworź stronę
2) otwórz stronę.html z wynikiem
3) informacje
4) backup danych
5) wyjdź

wybierz i wpisz odpowiedni nr, np: 1 lub 1.2:
|
```

- 1) Odpala kolejno: program szyfrujący, program deszyfrujący, program budujący stronę HTML i tworzy backup danych,
 - 1.1) Tylko szyfruje dane,
 - 1.2) Tylko deszyfruje dane,
 - 1.3) Tylko tworzy stronę HTML,
- 2) otwiera stronę HTML z wynikami w domyślnej przeglądarce, strona formatowana za pomocą pliku .css,
- 3) wyświetla krótki opis działania programu, tekst porabierany z pliku *info.txt*,
- 4) tworzy backup całego folderu (za wyjątkiem folderu *backup*) w folderze *backup/yyyy.mm.dd_hh.mm.ss (data_czas)*
- 5) zamyka program.

Każde polecenie zostaje opisane odpowiednim komunikatem, a następnie program oczekuje na wciśnięcie dowolnego klawisza, aby można było zapoznać się z komunikatem.

Struktura plików



Główny folder zawiera 3 głównie pliki:

- *szyfrowanie.exe* (szyfruje dane z pliku *Input/tekst.txt* i zapisujących w pliku *Output/Szyfr.txt*),
 - *deszyfrowanie.exe* (deszyfruje dane z pliku *Output/Szyfr.txt* i zapisuje w *Odszyfrowane.txt*)
 - *stronaHTML.exe* (tworzy stronę *stronaHTML/index.HTML* z danymi z plików *Output/Szyfr.txt* i *Odszyfrowane.txt*)
- oraz:
- *menu.bat* (, który był powyżej opisany)
 - *info.txt* (zawierający krótką informację o działaniu programu i wywoływany z menu).
- Zdecydowałem się podzielić główny program na 3 podprogramy, aby łatwiej było nimi zarządzać.

Backup - podfolder, w którym zapisywane są backupy (na rzucie powyżej pusty).

Input - podfolder, zawiera plik *tekst.txt*, który jest plikiem *wejściowym* tj. zawierającym tekst do zaszyfrowania.

Output - podfolder, zawiera pliki *wyjściowe*: *Szyfr.txt* (zawierający zaszyfrowane dane z *Input/Szyfr.txt*) oraz *Odszyfrowane.txt* (zawierającym odtworzoną wiadomość, deszyfrowaną z pliku *Szyfr.txt*)

pliki_py - podfolder zawierający pliki z .py w celu zachowania kodu oraz możliwości uruchomienia ich z pomocą pythona.

stronaHTML - podfolder, zawiera stronę *index.html* (z wynikami poleceń szyfrujących i deszyfrujących w postaci tabeli) oraz *style.css* (z formatowaniem tabeli do pliku *index.html*).

Implementacja

Szyfrowanie:

dane wejściowe:

str - ciąg znaków w postaci string

dane programu:

sys=2 (system kodowania)

tab=[] (pusta tablica)

program:

```
for i=0; i<długość(str); i++:  
    str[i] zamień na liczbę całkowitą (ASCII)  
    ord(str[i]) na system o podstawie sys  
    dołącz zakodowany znak do tab  
    sys= ord(str[i])%8+2  
zwróć tab
```

-ord() - funkcja zamieniająca znak na liczbę całkowitą na podstawie tablicy ASCII

Zamiana liczby dziesiętnej na inny system:

Dane wejściowe:

l - liczba do zakodowania

sys - nowy system liczbowy

Dane programu:

wynik - pusta tablica

program:

```
powtarzaj póki l != 0:  
    dołącz do wynik l (mod sys)  
    l=(l-(l%sys))/sys  
odwróć tablicę wynik  
zwróć wynik
```

Deszyfrowanie:

Dane wejściowe:

str - ciąg znaków (liczb) rozdzielonych spacją

Dane programu:

wynik, **tab_str** - puste tablica

sys = 2 (system liczbowy za pomocą którego deszyfrujemy znak)

suma = 0

Program:

przypisz do **tab_str** zmienną **str** podzieloną co spację (" ")

for **i**=0; **i**<długość(**tab_str**);**i**++:

 for **j**=0; **j**<długość(**tab_str**[**i**]);**j**++:

suma = **tab_str**[**i**] zamieniony na postać dziesiętną (z systemu o podstawie

sys)

sys = (**suma** (mod 8))+2

 dołącz do **wynik** chr(**suma**)

zwróć **wynik**

-chr() - zamienia postać całkowitą na znak (na podstawie tablicy ASCII)

-zamiana na postać dziesiętną następuje przez pomnożenie tej liczby poprzez **sys** do potęgi proporcjonalnej do jej miejsca (licząc od prawej)

-pętla ze zmienną **j** ma zastosowanie w realnym programie (przechodzi po kolejnych znakach zmiennej **tab_str**[**i**] (tablicy 2-wymiarowej)

Kody z plików:

menu.bat:

```
@echo off
@chcp 1250
attr 0F
:menu
cls
echo -----
echo  MENU
echo -----
echo.
echo 1) start programu
echo  1.1) tylko szyfruj
echo  1.2) tylko deszyfruj
echo  1.3) tylko tworź strone
echo 2) otwórz strone.html z wynikiem
echo 3) informacje
echo 4) backup danych
echo 5) wyjdź
echo.
set /p opcja=wybierz i wpisz odpowiedni nr, np: 1 lub 1.2:
if %opcja%==1 goto start
if %opcja%==1.1 goto szyfrowanie
if %opcja%==1.2 goto deszyfrowanie
if %opcja%==1.3 goto strona
if %opcja%==2 goto wyniki
if %opcja%==3 goto info
if %opcja%==4 goto backup
if %opcja%==5 exit
goto error

:start
cls
python szyfrowanie.exe
echo zaszyfrowano dane z Input/Tekst.txt do Output/Szyfr.txt
python deszyfrowanie.exe
echo zdeszyfrowano dane z Output/Szyfr.txt do Output/Deszyfrowane.txt
python stronaHTML.exe
echo utworzono strone HTML z wynikami powyższych poleceń (stronaHTML/index.html)
pause >null
goto backup

:szyfrowanie
```

```
python szyfrowanie.exe
echo zaszyfrowano dane z Input/Tekst.txt do Output/Szyfr.txt
pause >null
goto backup
```

```
:deszyfrowanie
python deszyfrowanie.exe
echo zdeszyfrowano dane z Output/Szyfr.txt do Output/Deszyfrowane.txt
pause >null
goto backup
```

```
:strona
python stronaHTML.exe
echo utworzono strone HTML z wynikami powyzzszych polecen (stronaHTML/index.html)
pause >null
goto backup
```

```
:wyniki
start stronaHTML/index.html
goto menu
:info
cls
type info.txt
pause >null
goto menu
```

```
:backup
cls
if not exist backup md backup
set
datestr=%date:~-4,4%.%date:~-7,2%.%date:~-10,2%_%time:~0,2%.%time:~3,2%.%time:~6,2%
md backup/%datestr%
robocopy ..\projekt_sem .\backup/%datestr% /S /XD backup
cls
echo stworzono kopie plikow w folderze backup/%datestr%
pause >nul
goto menu
```

```
:error
echo wpisano nieprawidlowe dane, wybierz sposrod: {1, 1.1, 1.2, 1.3, 2, 3 ,4}
pause >null
goto menu
```


szyfrowanie.py:

```
def DecToAny(num, sys):
    mod=[]
    while num!=0:
        mod.append(int(num % sys))
        num=int((num-(num % sys))/sys)
    mod.reverse()
    return("".join(str(x) for x in mod))

def szyfrowanie(str):
    tab=[]
    sys = 2
    for i in range(len(str)):
        tab.append(DecToAny(ord(str[i]), sys))
        sys = (ord(str[i]) % 8) + 2
    print(" ".join(tab))
    return(" ".join(tab))

def pelne_szyfrowanie(path_to_input, path_to_output):
    f_in = open(path_to_input,"r")
    open(path_to_output, 'w').close() #czyści plik przed zapisem, aby zapisać na czysto nowy
tekst
    f_out = open(path_to_output,"a")
    for l in f_in:
        f_out.write(szyfrowanie(l.rstrip())+'\n')
    f_in.close()
    f_out.close()

pelne_szyfrowanie("Input/tekst.txt","Output/Szyfr.txt")
```

deszyfrowanie.py:

#zamienia pojedynczą liniijkę str w zdeszyfrowany ciąg znakow

```
def deszyfrowanie(str):
```

```
    tab_str=str.split(" ")
```

```
    wynik=[]
```

```
    sys=2
```

```
    for i in range(len(tab_str)):
```

```
        suma=0
```

```
        dl=len(tab_str[i])
```

```
        for j in range(dl):
```

```
            suma+=int(tab_str[i][dl-1-j])*pow(sys,j)
```

```
        sys=(suma%8)+2
```

```
        wynik.append(chr(suma))
```

```
    return "".join(wynik)
```

#wczytuje linie dla 'deszyfrowanie()' z(path_to_input) i zapisuje(path_to_output)

```
def pelne_deszyfrowanie(path_to_input, path_to_output):
```

```
    f_in = open(path_to_input,"r")
```

```
    open(path_to_output, 'w').close() #czyści plik przed zapisem, aby zapisać na czysto nowy tekst
```

```
    f_out = open(path_to_output,"a")
```

```
    for l in f_in:
```

```
        f_out.write(deszyfrowanie(l.rstrip()))+"\n")
```

```
    f_in.close()
```

```
    f_out.close()
```

```
pelne_deszyfrowanie("Output/Szyfr.txt","Output/Odszyfrowane.txt")
```

stronaHTML.py:

```
def HTML_table_creator(file_1, file_2):
```

```
    table="<table class='table-fill'><tr><td class='text-left'>Zaszyfrowany tekst</td><td  
class='text-left'>odszyfrowany tekst</td></tr>\n"  
    for l, k in zip(file_1, file_2):  
        table = table+"<tr><td class='text-left'>" + l + "</td><td class='text-left'>" + k + "</td></tr>\n"  
    table=table+"</table>"  
    return table
```

```
def HTML_table(data_file1, data_file2, w_file):
```

```
    f_szyfr = open(data_file1, "r")  
    f_deszyfr = open(data_file2, "r")  
    open(w_file, "w").close()  
    f_zapis = open(w_file, "w")  
    f_zapis.write("<title>Wyniki</title>")  
    f_zapis.write("<div class='table-title'>")  
    f_zapis.write("<h3>Wyniki szyfrowania i deszyfrowania</h3>")  
    f_zapis.write("</div>")  
    f_zapis.write("<meta name='viewport' content='initial-scale=1.0; maximum-scale=1.0;  
width=device-width;'>\n")  
    f_zapis.write("<link rel='stylesheet' type='text/css' href='style.css'>\n")  
    f_zapis.write("<table class='table-fill'><tr><th class='text-left'>Zaszyfrowany tekst</th><th  
class='text-left'>Odszyfrowany tekst</th></tr>\n")  
    for l, k in zip(f_szyfr, f_deszyfr):  
        f_zapis.write("<tr><td class='text-left'>" + l + "</td><td class='text-left'>" + k + "</td></tr>\n")  
    f_zapis.write("</table>")  
    f_zapis.close()  
    f_szyfr.close()  
    f_deszyfr.close()
```

```
HTML_table("Output/Szyfr.txt", "Output/Odszyfrowane.txt", "stronaHTML/index.html")
```

