

Enigma - rozszerzona symulacja maszyny szyfrującej Enigma M3. Zawiera ona features, których fizyczna Enigma nie mogła posiadać ze względu na ograniczenia fizyczne i technologiczne. Pozwala ona na korzystanie z różnych alfabetów, a raczej zbiorów znaków, zawierających majuskuły, minuskuły, znaki interpunkcyjne i matematyczne, a nawet inne alfabety takie jak grecki, Cyrylica czy koreański. Pozwala ona nawet na korzystanie z dowolnej ilości wirników, jak i potrafi je sama stworzyć z danego jej alfabetu.

Opis implementacji:

Do projektu została użyta biblioteka *curses* (w całości do interface'u), kilka funkcji z biblioteki *os*, oraz po jednej funkcji z bibliotek *sys* i *random*.

Projekt został podzielony na dwa pliki z programem, zawierające 5 klas, oraz na liczne pliki konfiguracyjne w formacie tekstowym posegregowane na katalogi.

W pliku **simple.py** znajdują się cztery klasy:

- Klasa **Enigma** zawiera w sobie obiekt symulacji Enigmy, częścią której jest lista klas **Rotor** i klasa **Deflector** oraz zawiera metody kluczowe dla funkcjonalności programu:
 - **encrypt_sign** która wywołuje metody **encrypt_sign** i **decrypt_sign** z klasy **Rotor**, **encrypt_sign** z klasy **Deflector** oraz **add_shifts** z samej siebie
 - **add_shifts** która wywołuje **add_shift** z klasy **Rotor**, obracając wirniki zgodnie z zasadami działania Enigmy
 - **encrypt_key** która będąc swoistą nakładką na funkcję **encrypt_sign** upewnia się, że przekazany jej znak jest znakiem znajdującym się na wirnikach, jak i również dodaje go do tekstu jawnego i szyfrowanego
 - **encrypt_file** która otwiera plik, po czym wywołuje **encrypt_key** dla każdego kolejnego znaku w pliku, zwracając na koniec stworzony tym sposobem szyfr
 - **create_file** która zapisuje otrzymany tekst do pliku „output.txt”
- Klasa **Rotor** tworzy obiekt wirnika, który poza jego listą liter przetrzymuje także jego alfabet, ilość znaków w alfabecie, przesunięcie obecne i startowe oraz kilka kluczowych metod:
 - **encrypt_sign** zamieniającą dany znak z alfabetu na znak o jego pozycji w liście liter
 - **decrypt_sign** zamieniającą dany znak z listy liter na znak o jego pozycji w alfabecie
 - **add_shifts** „obracająca” ten wirnik o 1, zwracający **True** jeśli właśnie wystąpił na nim pełny obrót
- Klasa **Deflector** dziedziczy po klasie **Rotor** tworząc obiekt deflektora, który w Enigmie był tylko jeden i pozwalał na szyfrowanie i deszyfrowanie tym samym ustawieniem maszyny.
- Klasa **Create** która zawiera metody (najważniejszą z których jest **open_config**) otwierającą podane jej pliki konfiguracyjne i zwracającą z nich zmienne gotowe do zainicjowania klasy **Enigma**, jak i sprawdzenie uprzednio tych danych i wyrzucenie błędu informującego użytkownika co w jego plikach konfiguracyjnych jest nieprawidłowe, jeżeli jest cokolwiek

Uruchomienie tego pliku spowoduje zaszyfrowanie tekstu znajdującego się w „output.txt” domyślnymi ustawieniami i zapisanie go w tym samym „output.txt”

W pliku **main.py** znajduje się jedna klasa:

- Klasa **Initiate** pozwalająca wybrać pliki ustawień Enigmy, a nawet je stworzyć w programie. Zawiera wiele różnych metod obsługujących jedną z najważniejszych części programu - użytkownika. Najważniejsze z nich to:
 - Metoda **choose_config** pozwalającą użytkownikowi wybrać plik zawierający wirniki, stworzyć dowolną ilość własnych wirników korzystając z alfabetu z wybranego pliku bądź też wpisanego w programie, wykorzystując metody **clean_choose**, **select_name**, **choose_rotors**, **create_rotors**, **open_alphabet**, **create_alphabet** i **create_select**.
 - Metoda **choose_file** pozwalającą użytkownikowi czy chce zaszyfrować konkretny plik czy chce on wpisać tekst jawny w programie
 - Metoda **encrypt_input** pozwalającą użytkownikowi wpisać tekst jawny w programie, jednocześnie natychmiastowo pokazując mu zaszyfrowany tekst, który po jej zakończeniu zwraca w całości
 - Metoda **wrapped_functions**, która jest de facto funkcją **main** programu, ale musi być ona wywołana funkcją **curses.wrapper** w innej funkcji ze względu na to, jak funkcjonuje ekran w **curses**

Dołączony został również plik **main_raw.py**, który zawiera klasę **InitiateRaw**, zawierającą zmienione niektóre metody klasy **Initiate** tak, aby nie używać biblioteki **curses**. Zostało to dodane ze względu na fakt, iż domyślny terminal na systemie Windows nie jest w stanie używać tej biblioteki (dzięki czemu można go uruchomić na dowolnym systemie)

Opis konfiguracji:

Wszystkie pliki projektu są w plikach o rozszerzeniu „py” lub „txt”, lecz użytkownik może dodawać własne pliki w dowolnym prostym formacie tekstowym, dopóki nie posiada on dużej możliwości formatowania nie powinien być problemem dla programu.

Przykłady poszczególnych plików konfiguracyjnych zostały pokazane w instrukcji, tworząc własne pliki należy je naśladować, aby program był w stanie je poprawnie odczytać, oraz umieścić je w odpowiednim katalogu (który plik w którym katalogu jest również powiedziane w instrukcji).

Podsumowanie:

Wszystkie zamierzone rzeczy zostały wykonane, a nawet nadto. Jedynym problem może być fakt, iż w głównej wersji programu nie da się wpisywać znaków poza ASCII ze względu na to, jak działa biblioteka **curses**. Użytkownik jest jednak w stanie użyć głównej wersji z znakami spoza ASCII używając wczytywania z pliku. Jest on także w stanie wpisywać znaki spoza ASCII w programie uruchamiając wersję „raw”.

Program jest w stanie szyfrować używając dowolnej ilości wirników (sprawdzona poprawność na 99) oraz na dowolnym zestawie znaków poza spacją (sprawdzona poprawność na Greckim, Cyrylicy, Koreańskim, interpunkcji, liczbach, symbolach matematycznych oraz pełnym UTF-8)

W programie można samemu stworzyć plik z alfabetem, plik z wirnikami i deflektorami, plik konfiguracyjny **select** oraz samemu wpisać tekst jawny, dzięki czemu użytkownik nie musi tworzyć niczego sam poza programem martwiąc się o poprawność tego co zrobił. Jeśli jednak użytkownik zechce sam stworzyć plik konfiguracyjny i go użyć, program poinformuje go dokładnie co zapisał niepoprawnie, jeśli cokolwiek.

Źródła:

Informacje o tym jak działa Enigma:

- <https://www.youtube.com/watch?v=QwQVMqfoB2E>
- https://pl.wikipedia.org/wiki/Enigma#Opis_dzia%C5%82ania

Informacje o bibliotece curses:

- <https://docs.python.org/3/howto/curses.html>
- <https://docs.python.org/3/library/curses.html#module-curses>

Inne:

- <https://docs.python.org/3.7/library/random.html>
- <http://www.asciitable.com>
- <https://www.utf8-chartable.de>
- <https://stackoverflow.com/questions/704152/how-can-i-convert-a-character-to-a-integer-in-python-and-viceversa>
- <https://stackoverflow.com/questions/20774607/python-for-loop-inside-print/20774626>
- <https://stackoverflow.com/questions/2084508/clear-terminal-in-python>
- <https://www.guru99.com/reading-and-writing-files-in-python.html>