The Enigma Machine Simulator

Patryk J. Będkowski

Wydział Elektroniki i Technik Informacyjnych
Politechnika Warszawska

Styczeń 2021

Niniejszy projekt został utworzony w ramach przedmiotu Podstawy Informatyki i Programowania.

Spis Treści

I. Opis Projektu

Symulator Maszyny Szyfrującej Enigma jest programem, który symuluje działanie maszyny szyfrującej wykorzystywanej podczas Drugiej Wojny Światowej.

II. Wymagania Uruchomieniowe

Prawidłowe działanie programu wymaga zainstalowania **interpretera Python w wersji 3.6.**
Wymagane jest również zainstalowanie modułów używając komend:

```
pip -install pyqt5
pip -install tabulate
pip -install pyfiglet
```

bądź poprzez uruchomienie pliku `requirements.txt`, używając komendy:

```
pip install -r requirements.txt
```

Zaleca się utworzenie oddzielnego środowiska uruchomieniowego python.

III. Uruchomienie programu

W zależności od wyboru trybu pracy, program można uruchomić w dalej opisany sposób.
Symulator rozpoczyna pracę po wpisaniu w wierszu poleceń następującej komendy:

```
python enigma.py
```

Aby uruchomić program w trypie interfejsu graficznego, należy wprowadzić komendę:

```
python enigma_gui.py
```

IV. Opis Modułów

1. enigma_class

Zawiera implementację klasy `Enigma`, której metody używane są w programie. Wartości wstawiane do tej klasy są sprawdzane pod kątem ich poprawności. Jeżeli wprowadzona przez użytkownika wartość nie spełnia założeń programu, wyświetlany jest wyjątek ze stosownym komunikatem.

1.1. Enigma()

Obiekt klasy Enigma stanowi główną część programu. Przechowuje informacje o wprowadzonych ustawieniach. Posiada metody umożliwiające zapis wprowadzonych ustawień, sprawdzenia poprawności wprowadzonych danych.

Atrybuty `alpha, beta, gamma` musi posiadać typ `int`, jego wartość musi być w przedziale od 1 do 26. Ten warunek jest sprawdzany przez metodę `check_set_rotor_value().`

Atrybut `steckerbrett` musi posiadać typ `dict,` nie może przechowywać dwóch powtarzających się wartości, przechowywane wartości muszą należeć do alfabetu `ascii.` Te warunki są sprawdzane przez metody `steckerbrett_check_for_same_values()` oraz `steckerbrett_check_values().`

Atrybut `reflector` musi być jedną z wartości `A, B, C`. Ten warunek jest sprawdzany przez metodę `reflector_check_model().`

Szyfrowanie tekstu odbywa się po wywołaniu modułu `encryptingCodec` podając szyfrowany tekst jako jej argument.

2. enigma_interface

Moduł zawierający implementację klasy interfejsu `Enigma_interface()` odpowiadającej za interfejs użytkownika. o których warto wspomnieć

1.1. design_assumptions()

Metoda klasy interfejsu zwracająca założenia projektowe wprowadzanych wartości i ustawień. Każda pozycja przedstawia reguły uruchamianych ustawień niezbędnych do bezproblemowego działania symulatora.

1.2. start_menu()

Metoda wyświetlająca użytkownikowi dostępne opcje programu. Po wprowadzeniu numeru opcji przez użytkownika, następuje odwołanie się do modułu `setting_menu(),`

z którego zostają pobrane wprowadzone ustawienia symulatora. W dalszej kolejności następuje wywołanie metody `initiate_enigma_simulator().`

1.3. setting_menu()

Metoda, w której użytkownik określa sposób wprowadzenia ustawień do symulatora. Wprowadzone wartości zostają zwrócone

1.4. initiate_enigma_simulator()

Główna metoda, w której następuje proces odwołania się do klasy `Enigma` i wywołanie przetworzonego tekstu.
W przypadku wprowadzenia ustawień, które powodują wywołanie wyjątku, następuje ukazanie się odpowiedniego komunikatu użytkownikowi.
Użytkownik jest również proszony o ponowne wprowadzenie ustawień, jeżeli wprowadził je wcześniej ręcznie. Następuje odwołanie się do metody

1.5. insert_settings_by_hand()

Jeżeli ustawienia nie zostały wprowadzone przy pomocy pliku `.json`, program wyświetla komunikat o błędzie oraz prosi o zmianę błędnej zawartości z tego pliku, następuje zamknięcie programu.

Jeżeli nie wystąpi komunikat o błędzie i zostanie wyświetlony przetworzony tekst, użytkownik zostaje zapytany o możliwość utworzenia pliku z przetworzonym tekstem. Jeżeli zostanie przez niego wybrana opcja **`y,`** następuje odwołanie do metody

1.6. export_txt_menu().

Jeżeli użytkownik na początku działania programu zdecydował się wprowadzić ustawienia *ręcznie* , ma możliwość zapisania tych ustawień do pliku `.json.`

W przeciwnym wypadku nie zostaje wyświetlone zapytanie o utworzeniu pliku z ustawieniami.

Jeżeli zdecyduje się na zapis ustawień do pliku wpisując opcję **`y,`** następuje odwołanie do metody `export_json_menu().`

Program kończy działanie wyświetlając stosowny komunikat.

3. file_management

   Warstwa trwałości symulatora, obsługująca operacje odczytu i zapisu do pliku. Złożona jest z pięciu funkcji:

   1.1. check_if_ascii()

   sprawdza czy wartość jej argumentu znajduje się w alfabecie ascii.

   1.2. read_txt_file()

   jako argument przyjmuje ścieżkę do pliku tekstowego. Otwiera ten plik i pobiera z niego zawartość.
   Rzuca wyjątek jeżeli plik nie został odnaleziony, plik jest pusty, ilość wypełnionych linii jest różna od 1, znak/znaki w nim zawarte nie należą do alfabetu ascii.

   1.3. read_json_file()

   jako argument przyjmuje ścieżkę do pliku z rozszerzeniem .json. Otwiera ten plik i pobiera z niego zawartość. Rzuca wyjątek jeżeli plik nie został odnaleziony. Funkcja zwraca słownik zawierający ustawienia symulatora.

   1.4. save_json_file()

   jeżeli spełnione są wszystkie założenia, tworzy plik zawierający ustawienia, przyjmowane z argumentu, symulatora w formie słownika. Plik posiada nazwę wprowadzoną przez użytkownika za pomocą funkcji `input`. Rzuca wyjątek jeżeli nazwa nie została podana, nazwa pliku zawiera znaki inne od znaków alfabetu ascii oraz cyfr rzeczywistych.

4. enigma_gui

   Moduł uruchomieniowy programu w trybie interfejsu graficznego.

5. gui

   Moduł zawierający klasę `EnigmaUi()`, która odpowiedzialna jest za budowę i funkcjonowanie interfejsu graficznego.

6. enigma

   Główny moduł uruchomieniowy zawierający funkcję `main()`, za pomocą której następuje uruchomienie symulatora.

7. exceptions

   Moduł zawierający wyjątki dotyczące wprowadzanych wartości, sprawdzane w symulatorze.