AKADEMIA GÓRNICZO-HUTNICZA

Dokumentacja do projektu

Biblioteka do szyfrowania i deszyfrowania danych

z przedmiotu

Języki programowania obiektowego

EiT rok III

Szymon Rumin

piątek 9:45

prowadzący: mgr. inż. Jakub Zimnol

09.01.2025

1. Wprowadzenie

Projekt stanowi impementacje prostego, interaktywnego menu w konsoli umożliwiającego użytkownikowi korzystanie z dwóch różnych algorytmów szyfrowania i deszyfrowania danych takich jak: szyfr Cezara oraz szyfr płotkowy (rail fence cipher).

2. Zaimplementowane klasy

Encryptor

Abstrakcyjna klasa służąca jako fundament dla implementacji algorytmów szyfrowania: szyfr cezara oraz szyfr płotkowy. Posiada wirtualne metody, które są nadpisywane w klasach pochodnych.

Decryptor

Abstrakcyjna klasa służąca jako fundament dla implementacji algorytmów deszyfrowania: szyfr Cezara oraz szyfr płotkowy. Posiada wirtualne metody, które są nadpisywane w klasach pochodnych.

CaesarCipher

Posiada w sobie dwie klasy **CaesarEncryptor** oraz **CaesarDecryptor**, które dziedziczą odpowiednio po *Encryptor* oraz *Decryptor*.

Odpowiadają za szyfrowanie oraz deszyfrowanie tekstu za pomocą popularnego szyfru Cezara. Każda litera w tekście wejściowym jest przesuwana o określoną liczbę pozycji w alfabecie, która jest określana przez użytkownika, znaki alfabetyczne pozostają niezmienione.

Zawierają również metodę **description()**, która przechowuje szczegółowe informacje o zaszyfrowanym/deszyfrowanym tekście.

RailFenceCipher

Posiada w sobie dwie klasy **RailFenceEncryptor** oraz **RailFenceDecryptor,** które dziedziczą odpowiednio po *Encryptor* oraz *Decryptor*.

Ta pierwsza klasa odpowiada za szyfrowanie tekstu w układzie zygzakowatym na określonej liczbie szyn, którą użytkownik może ustalić, a następnie odczytaniu go wierszami, aby utworzyć zaszyfrowaną wiadomość.

Druga klasa pozwala na odszyfrowanie tekstu. Proces deszyfrowania polega na odtworzeniu układu zygzakowatego, aby odzyskać oryginalną wiadomość.

Klasy zawierają również metodę **description()**, która przechowuje szczegółowe informacje o zaszyfrowanym/deszyfrowanym tekście.

Menu

Klasa służąca do interakcji z użytkownikiem. Pozwala mu na zaszyfrowanie bądź deszyfrowanie danych i możliwość późniejszego zapisu do pliku. Obsługuje interakcje z użytkownikiem poprzez przejrzyste menu.

3. Opis uruchomienia

Projekt może być uruchomiony poprzez CMake oraz przy użyciu generatora MinGW Makefiles. Przed uruchomieniem należy się upewnić że posiadamy te narzędzia.

Uruchomienie projektu krok po kroku w konsoli:

git clone https://github.com/simrum03/JPO_Project.git

cd JPO_Project

mkdir build

cd build

cmake -G “MinGW Makefiles” ..

mingw32-make

./main

4. Przykładowe działanie programu

```
--------------------
Menu:
1 - Encrypting
2 - Decrypting
3 - Delete a file
4 - Quit program
--------------------
Enter your choice: 1
Encrypting selected. Choose an algorithm:
c - Caesar cipher
r - Rail fence cipher
b - Back to main menu
Enter your algorithm choice: c
Enter shift: 4
Enter data you would like to encrypt (type 'stop' to finish):
hello world
stop
Encrypted data: lipps asvph
What do you want to do with your encrypted data?
1 - Save data to a new file
2 - Append data to an existing file
3 - Do nothing
Enter your choice: 1
Enter filename: test.txt
Data has been saved to test.txt
Do you want to continue encrypting? (y/n): n

--------------------
Menu:
1 - Encrypting
2 - Decrypting
3 - Delete a file
4 - Quit program
--------------------
Enter your choice: 4
Quitting...
```