



AKADEMIA GÓRNICZO-HUTNICZA

Dokumentacja do projektu

Command-Line Cipher Application

z przedmiotu

Języki Programowania Obiektowego

Elektronika i Telekomunikacja, rok 3

Jan Rajski

Grupa 2, Środa, 9:45

prowadzący: mgr Jakub Zimnol

13.01.2026

Spis treści

| | | |
|------|---|---|
| 1. | Zastosowanie projektu | 3 |
| 2. | Struktura klas w projekcie..... | 3 |
| 3. | Opis klas zaimplementowanych w projekcie..... | 3 |
| 3.1. | Cipher..... | 3 |
| 3.2. | Caesar..... | 3 |
| 3.3. | ROT13..... | 3 |
| 3.4. | Atbash..... | 3 |
| 4. | Pliki pomocnicze..... | 3 |
| 4.1. | string_utilities.cpp..... | 3 |
| 4.2. | show_on_screen.cpp | 3 |
| 4.3. | app_logic.cpp | 3 |
| 5. | Kompilacja i uruchomienie aplikacji | 3 |
| 5.1. | Wymagania..... | 3 |
| 5.2. | Kompilacja..... | 4 |
| 5.3. | Uruchomienie..... | 4 |

1. Zastosowanie projektu

Projekt przedstawia 3 szyfry. Przy pomocy każdego z nich można zaszyfrować lub odszyfrować informacje (w formie zdań, słów), a następnie zapisać wynik, wraz z użytym kluczem, do pliku.

2. Struktura klas w projekcie

- Cipher
 - Caesar
 - ROT13
 - Atbash

3. Opis klas zaimplementowanych w projekcie

- 3.1. Cipher** – główna klasa projektu. Jest to klasa abstrakcyjna, będąca bazą dla innych, dziedziczących po niej klas. Posiada chronione pole *m_data* (wektor stringów), konstruktor parametryczny, metody *encode()* i *decode()* oraz wirtualny destruktor, ważny w celu przeciwdziałaniu wyciekowi pamięci.
- 3.2. Caesar** – klasa dziedzicząca po *Cipher*. Posiada konstruktor parametryczny, który wykorzystuje konstruktor z klasy *Cipher*. Ważnymi z punktu widzenia klasy są funkcje *key_check()* i *run_caesar_cipher()*. Zawiera także chronione pole *m_key* (integer), oraz odziedziczone i nadpisane metody *encode()* i *decode()*. Klasa jest wykorzystywana w celu użycia szyfru Cezara
- 3.3. ROT13** – klasa dziedzicząca po *Caesar*. Posiada konstruktor parametryczny, który wywołuje konstruktor *Caesar* z ustawionym na stałe kluczem. Klasa jest wykorzystywana w celu użycia szyfru *ROT13*.
- 3.4. Atbash** – klasa dziedzicząca po *Cipher*. Posiada konstruktor parametryczny, który wywołuje konstruktor *Cipher*. Zawiera także odziedziczone metody *encode()* i *decode()*. Klasa jest wykorzystywana w celu użycia szyfru *Atbash*.

W żadnej klasie nie zaimplementowano setterów i getterów, ponieważ do prawidłowego działania aplikacji są one zbędne. Podobnie z konstruktorami domyślnymi – użycie ich uniemożliwiłoby poprawne zainicjalizowanie danych.

4. Pliki pomocnicze

- 4.1. string_utilities.cpp** – plik zawierający funkcje wykonujące działania na ciągach znaków.
- 4.2. show_on_screen.cpp** – plik zawierający funkcje wyświetlające informacje dla użytkownika.
- 4.3. app_logic.cpp** – plik zawierający główną logikę działania aplikacji (m.in. polimorfizm).

5. Kompilacja i uruchomienie aplikacji

5.1. Wymagania

- Cmake w wersji 3.11 lub wyższej

- Kompilator obsługujący C++ 11 lub wyższy (np. g++ 4.8.1)

5.2. Kompilacja

- Wejść do głównego katalogu z projektem i utworzyć folder *build*.
- W konsoli, będąc w folderze *build* wpisać: *cmake ..*
- Następnie: *cmake --build .*

5.3. Uruchomienie

- Będąc dalej w katalogu build wpisać *.\Command-Line-Cipher-Application.exe*