Politechnika Śląska

Wydział Matematyk Stosowanej

Kierunek Informatyka

Gliwice, 01.10.2020

Programowanie I

**projekt zaliczeniowy**

**"*Maszyna szyfrująca*"**

**Natalia Staszak gr. lab. 3/6**

**1. Opis projektu.**

Aplikacja konsolowa - Symulator maszyny szyfrującej. Użytkownik wybiera jeden z dostępnych szyfrów, następnie wprowadza z klawiatury wiadomość(input) oraz ewentualnie inne potrzebne dane. Program dokonuje procesu szyfrowania i zwraca output - zaszyfrowaną wiadomość

**2. Wymagania**

1. Pobranie inputu od użytkownika
2. Wybór szyfru przez użytkownika (ad 6.)
3. Możliwość uzyskania skrótowej informacji na temat wybranego szyfru
4. Uruchomienie odpowiedniej funkcji szyfrującej
5. Zwrot przekształconego inputu
6. Dostępne funkcje:
   1. szyfr Cezara;
   2. kodowanie RSA;
   3. Alfabet Morsa;
   4. szyfr Atbasz;
   5. szyfr podstawieniowy z wykorzystaniem nietypowych znaków ASCII

**3. Przebieg realizacji**

Program składa się z trzech plików

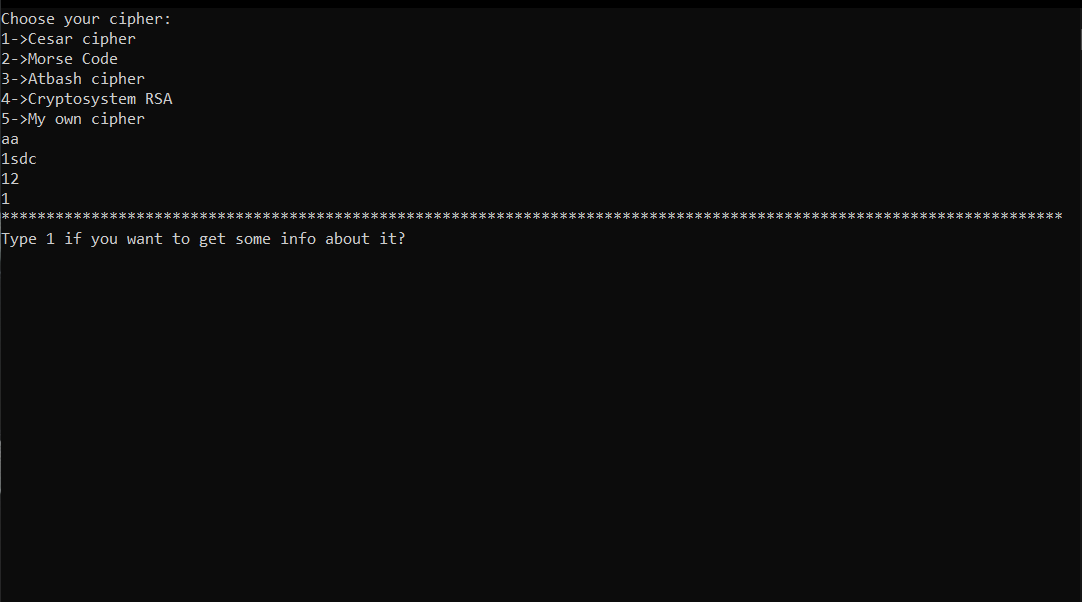
1. header.h - dołączenie wszystkich bibliotek i deklaracja funkcji
2. funkcje.cpp - definicje funkcji głównych(szyfry ad.2.6) info i pomocniczych
3. main.cpp - szereg instrukcji tworzących menu interakcji z użytkownikiem i wywołanie odpowiednich funkcji

Funkcje:

1. Funkcje główne:
   1. **void Morse() -** funkcja pobiera od użytkownika wiadomość do zmiennej *text* następnie za pomocą metody *std::transform(text.begin(), text.end(),text.begin(), ::toupper);* zamienia całość na wielkie litery następnie iteruje po wiadomości i rozpoznaje kolejne znaki i przerabia je na kod Morse’a, przyrównując do siebie listę *char alphabet* zawierającą 26 liter alfabetu łacińskiego, znak spacji oraz cyfry od zera do dziewięć i listę *string code* zawierającą odwzorowanie poszczególnych znaków w kodzie Morse’a (indeks każdego znaku w liście *alphabet* odpowiada indeksowi jego odpowiednika w liście *code*.) Następnie funkcja drukuje na ekranie otrzymaną zaszyfrowaną sekwencję
   2. **void Cesar() -** funkcja pobiera od użytkownika wiadomość do zmiennej *text* i klucz(ilość miejsc przy przestawieniu) do zmiennej key. Następnie iteruje po wiadomości, sprawdza czy dana litera jest duża czy mała i odpowiednio przesuwa ją o wartość key. Następnie drukuje zmienioną wiadomość.
   3. **void Atbash() -** funkcja pobiera od użytkownika wiadomość do zmiennej *text.* Następnie iteruje po wiadomości, sprawdza czy dana litera jest duża czy mała (w tym celu korzysta z funkcji *int check()*( ad.2.a) i odpowiednio przesuwa ją o odpowiadającą jej wartość.(literze znajdującej się na X’owym miejscu w alfabecie odpowiada litera znajdująca się na X’owym miejscu od końca) Następnie drukuje zmienioną wiadomość.
   4. **void RSA() -** funkcja pobiera od użytkownika początek i koniec przedziału, z którego losowane mają być liczby (za pomocą funkcji *bool is\_prime()(ad.2.b) i int random\_prime()(ad.2.c)*) do zmiennych *p* i *q* i klucz prywatny do zmiennej *e.* Następnie dokonuje szeregu obliczeń niezbędnych do uzyskania klucza prywatnego zapisanego w zmiennej *k* i drugiego elementu klucza publicznego zapisanego w zmiennej *n*. Następnie iteruje po wiadomości i szyfruje każdy znak podnosząc jego wartość liczbową modulo *n* do potęgi równej *e.* Następnie funkcja drukuje sekwencję zaszyfrowanych liczb, klucz publiczny *e,n* i klucz prywatny *k*.
   5. **void My\_own\_cipher() -** funkcja pobiera od użytkownika wiadomość do zmiennej *text* następnie za pomocą metody *std::transform(text.begin(), text.end(),text.begin(), ::toupper);* zamienia całość na wielkie litery następnie iteruje po wiadomości i rozpoznaje kolejne znaki i przerabia je na szyfr, przyrównując do siebie listę *char alphabet* zawierającą 26 liter alfabetu łacińskiego, i listę *char code* zawierającą odwzorowanie poszczególnych znaków w za pomocą nietypowych znaków ASCII jak strzałki czy emotikony (indeks każdego znaku w liście *alphabet* odpowiada indeksowi jego odpowiednika w liście *code*.) Następnie funkcja drukuje na ekranie otrzymaną zaszyfrowaną sekwencję.
2. Funkcje pomocnicze:
   1. **int check(char c) -** funkcjasprawdza czy dany znak c jest wielką czy małą literą( czy mieści się w odpowiednim przedziale liczbowym odpowiadającym danym literą w kodzie ASCII) następnie zwraca wartości odpowiednio 1 i 0, jeżeli znak nie jest literą zwraca 2
   2. **bool is\_prime(int n) -** funkcja sprawdza czy dana liczba n jest liczbą pierwszą. Funkcja sprawdza przedział od 2 do pierwiastek z n w poszukiwaniu dzielników liczby n(reszta z dzielenia n przez tą liczbę wynosi zero) jeśli taki znajdzie zwraca wartość *false,* jeśli pętla się wykona do końca zwraca wartość *true.*
   3. **int random\_prime (int a, int b) -** funkcja za pomocą generatora liczb pseudolosowych *mt19937* losuje liczbę z przedziału (a,b) i zapisuje ją do zmiennej x. Operacja wykonywana jest tak długo, aż wartość x będzie liczbą pierwszą (funkcja *bool is\_prime()*(ad.2.b))
3. Funkcje info:
   1. **void \*\*\*\_info()** - drukuje krótki opis histori i działania danego szyfru.
4. Zaimportowane biblioteki:
   1. **<iostream> -** umożliwia wykonywanie operacji na standardowych strumieniach wejścia/wyjścia.
   2. **<string> -** posiada zaimplementowaną uogólnioną klasę napisów zwaną string, ułatwienie pracy na tekstach.
   3. **<math.h> -** biblioteka zawierająca funkcje matematyczne np. pierwiastek (*sqrt)*
   4. **<random> -** biblioteka zawierająca generator liczb pseudolosowych
   5. **<algorithm>** biblioteka zawierająca różne algorytmy i metody (przekształcanie na duże litery)
5. main:
   1. program wyświetla w konsoli dostępne szyfry i przypisane im wartości od 1 do 5
   2. Następnie pobiera od użytkownika input zapisując go do zmiennej *choice*.
   3. Jeżeli dane nie zostały wprowadzone w sposób poprawny czeka na kolejny input.
   4. Następnie program proponuje użytkownikowi otrzymanie informacji o wybranym szyfrze poprzez wybranie liczby 1
   5. Program pobiera kolejny input do zmiennej *info* i sprawdza czy spełnia powyższy warunek. Jeżeli tak wywołana zostaje odpowiednia funkcja *void info()*
   6. Następnie wywołuje odpowiednią funkcję szyfrującą
   7. Program napisany jest całkowicie w języku angielskim (nie licząc dokumentacji i komentarzy w kodzie), za pomocą języka C++,
   8. Program dostosowany jest do angielskiego alfabetu łacińskiego.

**4. Instrukcja użytkownika**

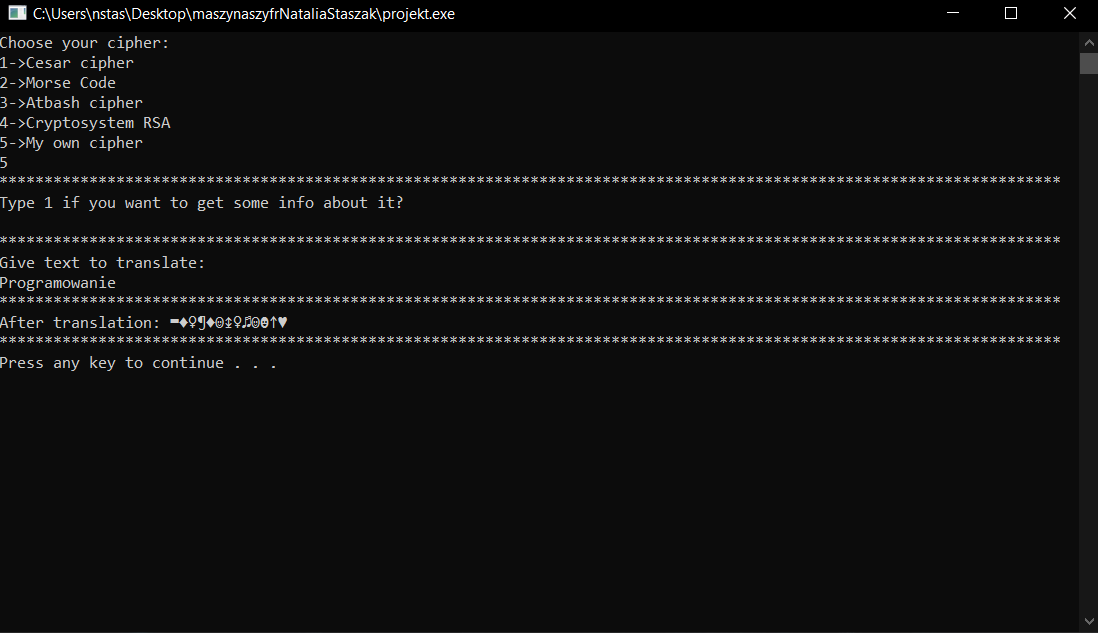
Program wyświetla w konsoli dostępne szyfry i przypisane im wartości od 1 do 5. Jeżeli dane nie zostały wprowadzone w sposób poprawny czeka na kolejny input.



Następnie program proponuje użytkownikowi otrzymanie informacji o wybranym szyfrze poprzez wybranie liczby 1. Jeżeli użytkownik wyrazi chęć, wywołana zostaje odpowiednia funkcja



Następnie wywołuje odpowiednią funkcję szyfrującą



**5. Podsumowanie i wnioski.**

Zrealizowano wszystkie podstawowe założenia. Istotny problem w czasie realizacji projektu stanowił algorytm RSA (np. potęgowanie dużych liczb modulo n), który w finalnej wersji pozostaje niedoskonały (e podaje użytkownik). Problem stanowiły polskie znaki w szyfrze Cezara i Atbash, więc program został dostosowany do języka angielskiego by uniknąć niekonsekwencji.