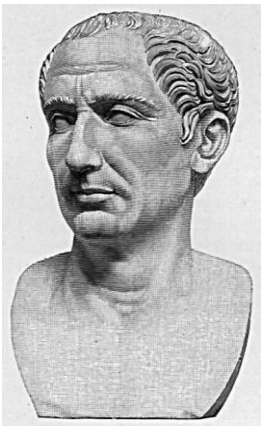
Algorytm Szyfru Cezara



Szyfr Cezara to jedna z najprostszych technik szyfrowania. Nazwa szyfru pochodzi od Juliusza Cezara, który używał tej techniki do komunikacji ze swymi przyjaciółmi.

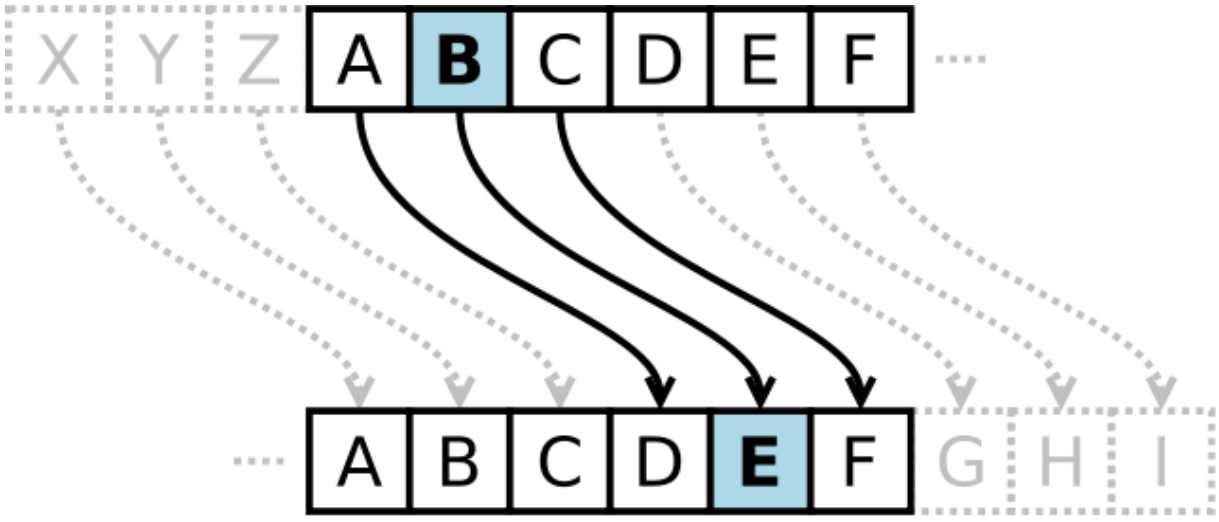
Jak podaje starożytny pisarz Swetoniusz robił to zamieniając dana literę znakiem znajdującym się 3 pozycje dalej w alfabecie:

Gdy miał cokolwiek poufnego do powiedzenia, pisał to szyfrem. Zmieniał daną literę na inną, tak by zdania nie przypominały znanych słów. Jeśli ktokolwiek chciał to odszyfrować i zrozumieć znaczenie wiadomości, musiał zamienić czwartą literę (D) na A i tak samo z kolejnymi.

Swetoniusz „Żywot Juliusza Cezara” 56

Szyfr Cezara

W *Szyfrze Cezara* każdą literę alfabetu zamieniamy na literę znajdującą się 3 pozycje dalej.



Zamiana wygląda następująco: A → D, B → E, C → F, itd. Dla ostatnich znaków X → A, Y → B, Z → C.

Właściwości szyfru Cezara

Jest to rodzaj szyfru podstawieniowego, w którym każda litera tekstu jawnego (niezaszyfrowanego) zastępowana jest inną, oddaloną od niej o stałą liczbę pozycji w alfabecie, literą (szyfr monoalfabetyczny), przy czym kierunek zamiany musi być zachowany. Nie rozróżnia się przy tym liter dużych i małych.

Jest to szyfr przesuwający z kluczem 3. Ten szczególny przypadek jest określany jako szyfr Cezara, a sam ***termin szyfr przesuwający*** jest zarezerwowany dla przypadku ogólnego.

Kodowanie w językach programowania

Szyfr Cezara jest prosty i łatwym do implementacji algorytmem.

Można skorzystać co najmniej z trzech sposobów implementacji:

Użycie polecenia IF

Algorytm będzie miał postać:

if `znak_jawny = 'a'` **then** `znak_zaszyfrowany` → `'d'`

if `znak_jawny = 'b'` **then** `znak_zaszyfrowany` → `'e'`

…

Odszyfrowanie będzie miało postać odwrotną:

if `znak_zaszyfrowany = 'd'` **then** `znak_jawny` → `'a'`

if `znak_zaszyfrowany = 'e'` **then** `znak_jawny` → `'b'`

…

Zaletą jest prostota algorytmu nie wymagająca użycia i znajomości żadnych skomplikowanych konstrukcji i poleceń.

Wadą jest duży nakład pracy i problemy z każdą zmianą klucza przesuwającego.

Użycie tablicy 2 – wymiarowej

Postać algorytmu:

Algorytm korzysta z tablicy 2 wymiarowej mającej 2 wiersze i liczbę kolumn zależną od długości alfabetu danego języka.

X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z

Przy szyfrowaniu kolejnych znaków tekstu jawnego, należy sprawdzić w której pozycji pierwszego wiersza znajduje się dany znak i zamiast niego wstawić literę z tej samej kolumny ale wiersza poniżej.

K → N
O → R
T → W

Przy deszyfrowaniu sprawdzamy znaki z dolnego wiersza. Gdy w tekście zaszyfrowanym pojawia się odpowiedni znak, należy go poszukać w dolnym wierszu. Po znalezieniu zamiast niego trzeba wstawić literę z tej samej kolumny ale wiersza powyżej.

N → K
R → O
W → T

Zaletą jest prostota i elegancja algorytmu. Łatwo też zmodyfikować ustawienia znaków w tablicy. Można też dodać inne znaki do podstawowego alfabetu.

Wadą jest duża ilość operacji. Program musi za każdym razem przeszukać całą tablicę i sprawdzać gdzie dany znak się znajduje. W ten sposób czas operacji to iloczyn długości tekstu i długości tablicy ze znakami. Dla języka angielskiego algorytm działa 26 razy dłużej.

Kodowanie ASCII

W tym algorytmie wykorzystuje się sposób kodowania liter, cyfr i znaków interpunkcyjnych za pomocą liczb. Podstawowy sposób kodowania nazywa się ASCII i obejmuje zakres od 0 do 127. W tej części są zawarty duże (od 65 do 90) i małe litery (od 97 do 122).

Działanie algorytmu:

Przy szyfrowaniu kolejnych znaków należy obliczyć ich kod ASCII.

```
ord (x)
```

Następnie dana liczba jest zwiększana o 3.

```
ord (x) + 3
```

Potem wystarczy zamienić to na znak i litera jest zaszyfrowana.

```
char (ord (x) + 3)
```

Zapętlenie algorytmu:

Niestety pojawia się problem z ostatnimi literami. Po dodaniu do nich 3, nie pokazują początku alfabetu, ale znaki znajdujące się dalej. Konieczne jest zapętlenie i zmuszenie, by po dojściu do ostatniej litery wracał do początku alfabetu.

Do tego celu służy dzielenie modulo o długości równej ilości znaków alfabetu.

```
char ((ord (x) + 3) mod 26)
```

Niestety nie pozwala to na łatwe uzyskanie rozwiązania. Dzielenie przesunie wynik w zakres 0 – 26, w którym nie ma liter tylko znaki sterujące. Konieczne jest przesunięcie całego zbioru liter w tą część tablicy ASCII, a potem powrót do poprzedniego zakresu.

Dla małych liter (z zakresu 97- 122) będzie to wyglądać następująco:

```
char (((ord (x)- 97 + 3) mod 26)+ 97)
```

Algorytm deszyfrujący:

Szyfr Cezara jest szyfrem symetrycznym. Wystarczy tylko zmienić kolejność przesunięcia zastępowanych liter. Zamiast w prawo – lewo. W omawianym przykładzie zamiast **+ 3** będzie **-3**.

```
char (((ord (x)- 97 - 3) mod 26)+ 97)
```

Zaletą algorytmu jest prostota i elegancja. Bardzo łatwo zmodyfikować klucz szyfrujący.

Wadą jest dość rozbudowany operand zamiany. Konieczne jest też oddzielne ustawienie zakresów dla małych i dużych liczb (jeśli są używane w tekście).

Kryptoanaliza:

Możliwe jest odczytanie wiadomości mając sam tekst zaszyfrowany (bez znajomości klucza).

Metoda brute-force

Najpopularniejszym i najprostszym sposobem jest sprawdzenie wszystkich wariantów klucza. Jest ich niewiele – o 1 mniej niż liter w alfabecie. Można to zrobić łatwo za pomocą pętli.

```
for i = 0 to n-1
      char (((ord (x)- 97 - i) mod 26)+ 97)
```

Wynikiem jest lista różnych odpowiedzi, z których trzeba wybrać tę zrozumiałą (zapisaną w języku naturalnym).

Metoda statystyczna

Inna metodą jest wykorzystanie statystycznych cech języka. W każdym języku częstotliwość występowania poszczególnych liter jest inna. Mając tekst zaszyfrowany można policzyć ile razy dany znak występuje i dopasować go do rozkładu częstości liter.

Metoda ta jednak wymaga znajomości w jakim języku jest napisany tekst wiadomości, gdyż rozkłady są różne dla poszczególnych języków. Potrzebne jest też zgromadzenie większej ilości zaszyfrowanych wiadomości, gdyż dopiero wtedy cechy statystyczne języka staną się wyraźne.

Ćwiczenia

1. Używając szyfru Cezara napisz program:
    a. szyfrujący wiadomość
    b. deszyfrujący wiadomość
2. Rozbuduj program dodając możliwość zmiany klucza o dowolną wartość.
3. Napisz program szyfrujący Cezarem, używając tablicy 1-wymiarowej. Zastosuj szyfrowanie poprzez przesunięcie indeksu tablicy o 3.
4. Napisz program szyfrujący Cezarem, który pozwala na użycie pełnego alfabetu polskiego (ą, ę, ć, ń, ł, ó, ś, z, ź). Zastosuj tablicę lub inną strukturę danych.
5. Napisz program odczytujący zaszyfrowaną szyfrem Cezara wiadomość, używając metody brute-force
6. Napisz program odczytujący zaszyfrowaną szyfrem Cezara wiadomość, używając metod statystycznych.