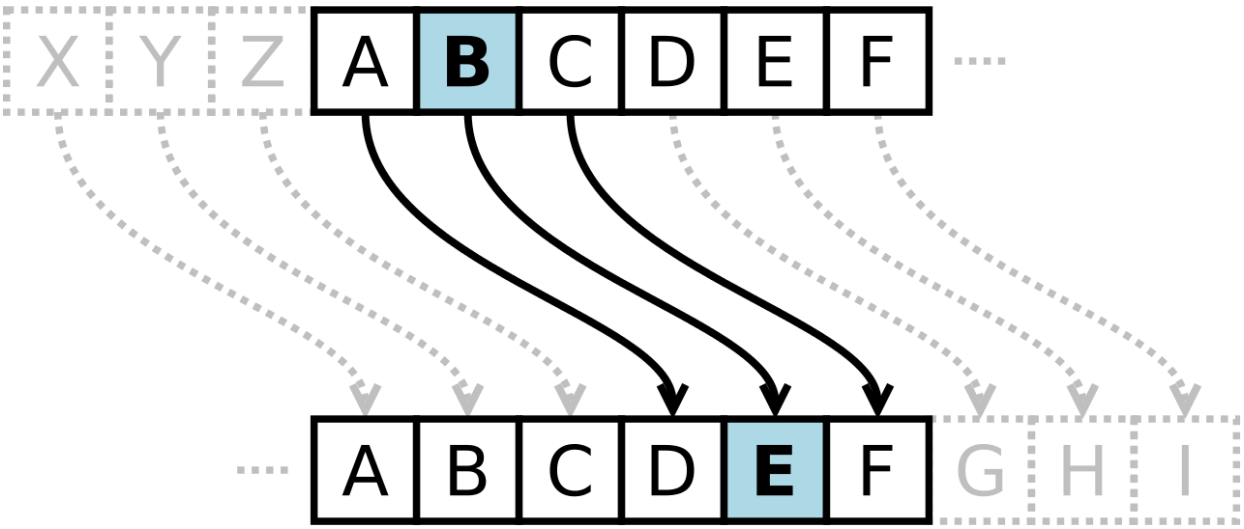
1. **Wprowadzenie**

Nazwa szyfru Cezara pochodzi od rzymskiego wodza i polityka Juliusza Cezara, który szyfrował swoją korespondencję poprzez podmianę każdej litery wiadomości na taką znajdującą się w alfabecie o 3 pozycje dalej. Dzisiejszy szyfr Cezara jest uogólnieniem tego rozwiązania - opiera się on na zastąpieniu litery inną literą odległą o n pozycji, przy zachowaniu kierunku zmiany dla całego tekstu. Stanowi rodzaj szyfru podstawieniowego. Jest to jedna z prostszych metod szyfrowania, niegwarantująca tajności komunikacji.

2. **Opis działania**

Działanie algorytmu polega na zastąpieniu litery x literą y, oddaloną o n miejsc w alfabecie. Implementacja w języku python3 oparta jest na funkcjach ord - zwracającej kod danego znaku (litery) i chr - zamieniającej dany kod na znak. Funkcja operuje na alfabecie ‘abcdefghijklmnopqrstuvwxyz’, gdzie ord(‘a’) = 97 i ord(‘z’) = 122. W wypadku gdy kod zaszyfrowanego znaku nie mieści się w przedziale [97;122] odejmowana jest długość alfabetu (26), gdy kod znaku jest większy od 122 i dodawana długość alfabetu, gdy kod znaku jest mniejszy od 97 (wyjątek ten ma miejsce przy rozszyfrowywaniu). Aby rozszyfrować dany zaszyfrowany tekst, znając klucz, należy do funkcji przekazać zaszyfrowany tekst i liczbę przeciwną do klucza, tj. Iloczyn klucza i liczby –1. Algorytm jest o złożoności liniowej.

3. **Wizualizacja**



https://pl.wikipedia.org/wiki/Szyfr_Cezara#/media/Plik:Caesar3.svg

Grafika obrazująca szyfr Cezara z kluczem n=3

4. Przykładowy kod w języku Python3

```
def szyfr(text, n):
    """Funkcja szyfrująca za pomocą szyfru Cezara wiadomości zapisane znakami w alfabecie abcdefghijklmnopqrstuvwxyz"""
    n = n % 26 # Reszta z dzielenia przez długość alfabetu to rzeczywiste przesunięcie liter
    ret = '' # Wynik działania algorytmu
    for litera in text:
        if ord(litera) + n > 122: # Jeżeli nowa liczba przekracza zakres alfabetu z prawej strony
            ret += chr(ord(litera) + n - 26) # Dodawanie zaszyfrowanego znaku do wyniku
        elif ord(litera) + n < 97: # Pętla potrzebna przy rozszyfrowywaniu, gdy nowa liczba przekracza zakres alfabetu z lewej strony
            ret += chr(ord(litera) + n + 26) # Dodawanie zaszyfrowanego znaku do wyniku
        else: # Gdy nowy kod mieści się w zakresie alfabetu
            ret += chr(ord(litera) + n) # Dodawanie zaszyfrowanego znaku do wyniku
    return ret
```

5. Inna implementacja algorytmu

Implementacja algorytmu przyjmuje jako jeden z argumentów alfabet, w którym zaszyfrowana jest wiadomość. Opiera się na tych samych zasadach, co wcześniejsze rozwiązanie, lecz dzięki zastosowaniu indexów z zakresu [0;długość alfabetu) unikamy przypadków skrajnych związanych z kodami znaków a-z zwracanych przez funkcję ord.

```
def szyfr_2(alfabet, text, n):
    """Funkcja szyfrująca za pomocą szyfru Cezara wiadomości zapisane znakami w podanym alfabecie """
    n = n % len(alfabet) # Reszta z dzielenia przez długość alfabetu to rzeczywiste przesunięcie liter
    ret = ''
    for litera in text:
        if alfabet.index(litera) + n > len(alfabet):
            ret += alfabet[alfabet.index(litera) + n - len(alfabet)]
        else:
            ret += alfabet[alfabet.index(litera) + n]
    return ret
```