# Esercizio guidato

Settimana 8
22/11/2022

Si ringrazia il Dott. Giacomo Baruzzo per il materiale

# Crittografia: definizione

La crittografia è un campo di ricerca che si occupa della protezione delle informazioni.

La protezione delle informazioni è effettuata tramite la loro conversione in formati che non possono essere comprensibili a utenti non autorizzati.

*Dove troviamo la crittografia?*

- E-commerce
- Password
- Comunicazioni militari
- Transazioni bancarie
- Spionaggio
- …

# Crittografia: concetti base

**Testo in chiaro**: informazione che si vuole proteggere; solitamente è in un formato comprensibile

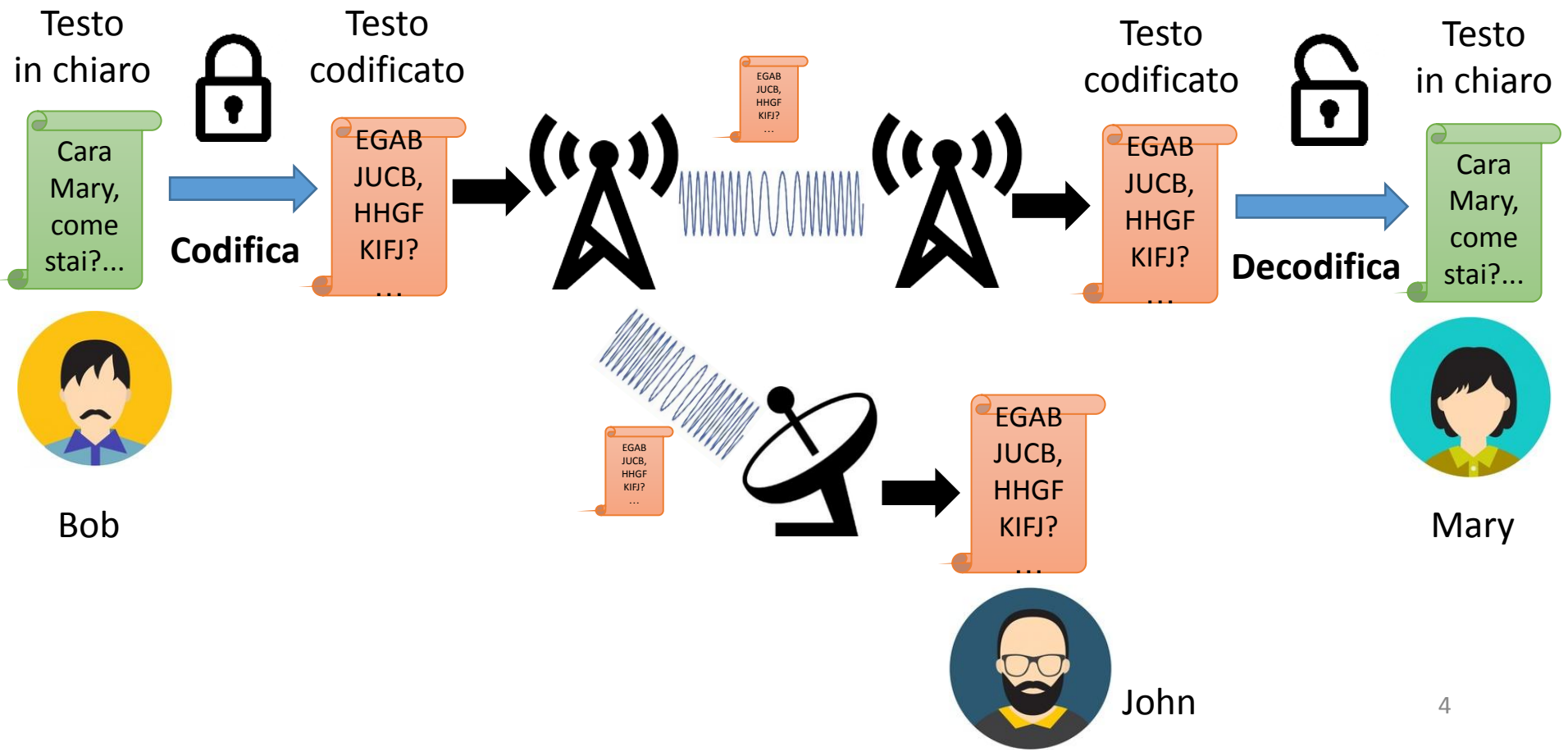
**Testo codificato/cifrato**: risultato della trasformazione da testo in chiaro ad un formato non comprensibile ad utenti non autorizzati

**Codifica/cifratura**: processo di trasformazione da testo in chiaro a testo codificato

**Decodifica/decifratura**: processo di trasformazione da testo codificato a testo in chiaro

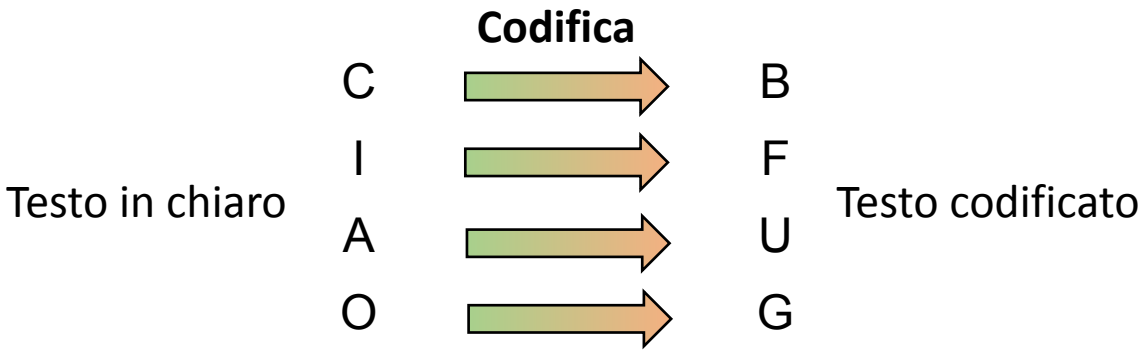
# Crittografia: esempio

Bob vuole inviare un messaggio a Mary, senza che John possa leggerlo…

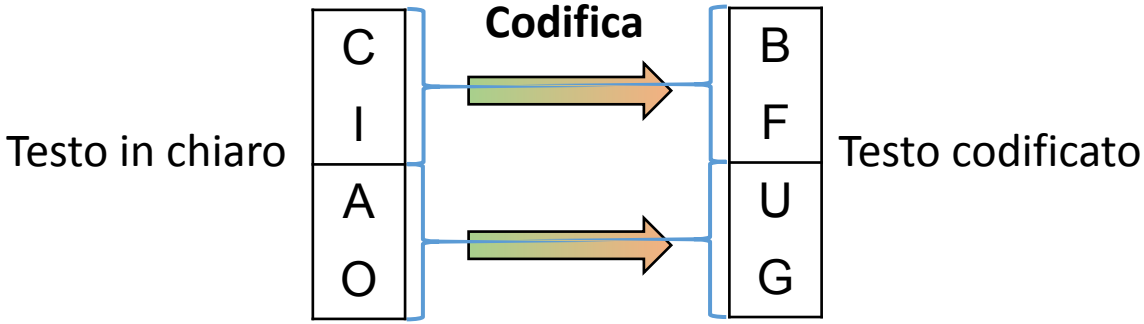


# Metodi di codifica

- Codifica a singolo carattere: la codifica viene applicata singolarmente su ogni carattere del testo




- Codifica a coppie di caratteri: la codifica viene applicata su coppie di caratteri



# Metodi di codifica

- I metodi di codifica appena visti sono tra i più semplici e quelli che vedrete nelle esercitazioni
- Non sono utilizzati nelle applicazioni reali
- Esistono metodi di codifica più complessi (e più “sicuri”), che non vedremo in laboratorio

L’esercizio di oggi riguarda la cifratura di Cesare, un metodo di codifica a **singolo carattere**.

Nelle prossime slide, presenteremo la soluzione dell’esercizio, focalizzandoci solo sull’implementazione delle funzioni di **codifica/decodifica** di un singolo carattere.

# Codifica di Cesare

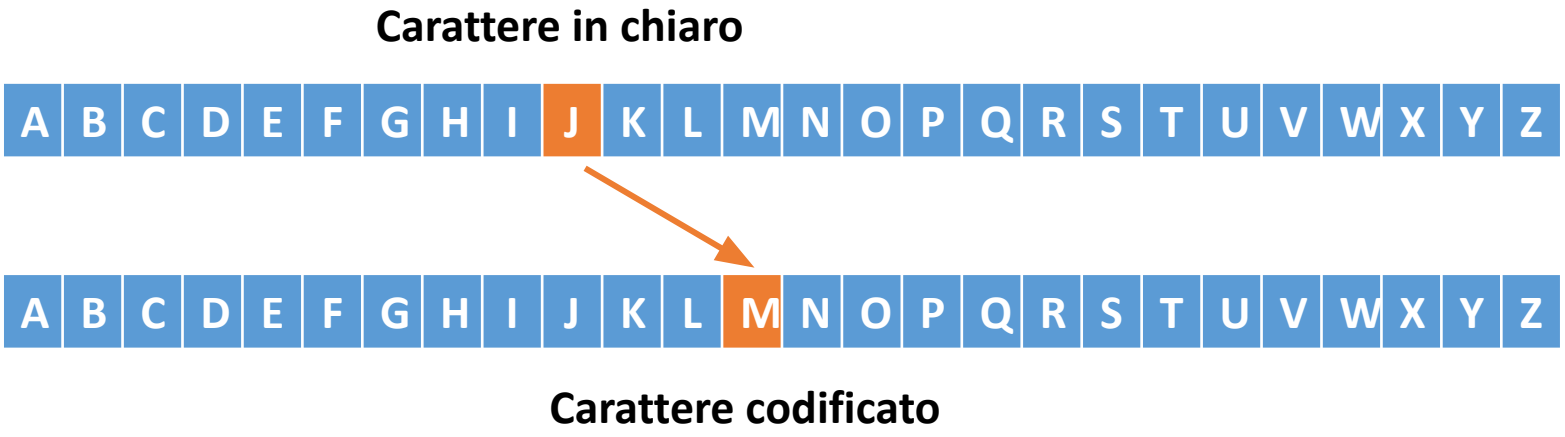
La cifratura di Cesare opera sostituendo ciascuna lettera del testo (i caratteri che non sono lettere rimangono identici nel testo cifrato) con un'altra lettera determinata procedendo "in avanti" nell'alfabeto di un numero di posti uguale a PARAM, ripartendo dall'inizio se si arriva all'ultima lettera.

Ad esempio, con PARAM=3, la lettera **E** viene sostituita dalla lettera **H**

# Codifica di Cesare

La cifratura di Cesare opera sostituendo ciascuna lettera del testo (i caratteri che non sono lettere rimangono identici nel testo cifrato) con un'altra lettera determinata procedendo "in avanti" nell'alfabeto di un numero di posti uguale a PARAM, ripartendo dall'inizio se si arriva all'ultima lettera.

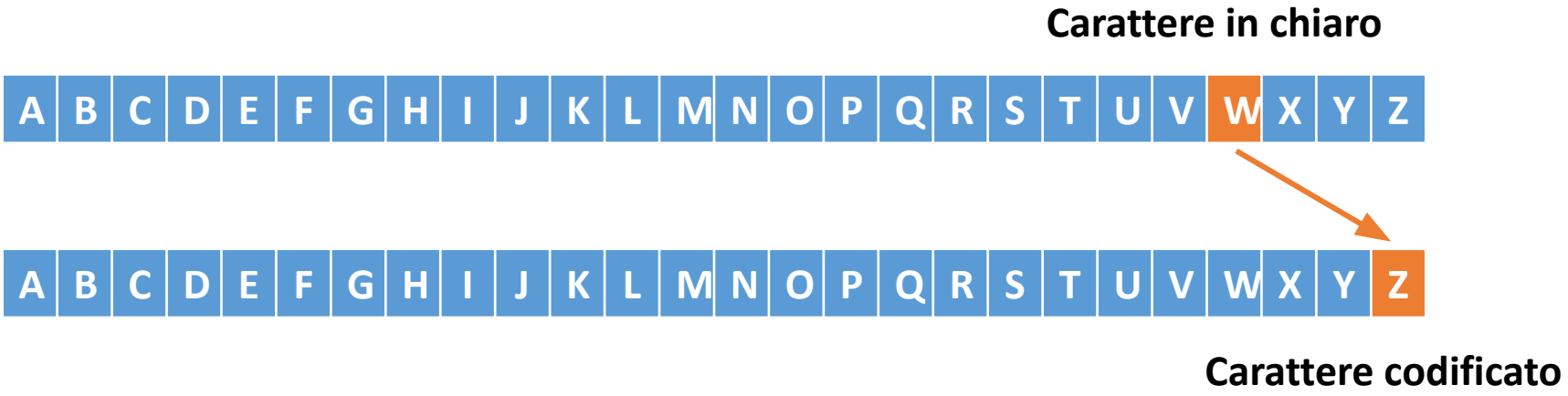
Ad esempio, con PARAM=3, la lettera E viene sostituita dalla lettera H, **J** dalla **M**



# Codifica di Cesare

La cifratura di Cesare opera sostituendo ciascuna lettera del testo (i caratteri che non sono lettere rimangono identici nel testo cifrato) con un'altra lettera determinata procedendo "in avanti" nell'alfabeto di un numero di posti uguale a PARAM, ripartendo dall'inizio se si arriva all'ultima lettera.

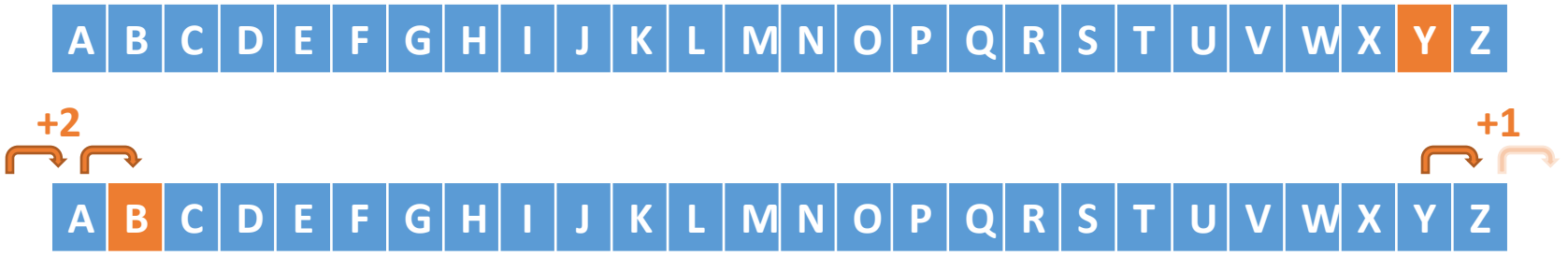
Ad esempio, con PARAM=3, la lettera E viene sostituita dalla lettera H, J dalla M, **W** dalla **Z**



# Codifica di Cesare

La cifratura di Cesare opera sostituendo ciascuna lettera del testo (i caratteri che non sono lettere rimangono identici nel testo cifrato) con un'altra lettera determinata procedendo "in avanti" nell'alfabeto di un numero di posti uguale a PARAM, ripartendo dall'inizio se si arriva all'ultima lettera.

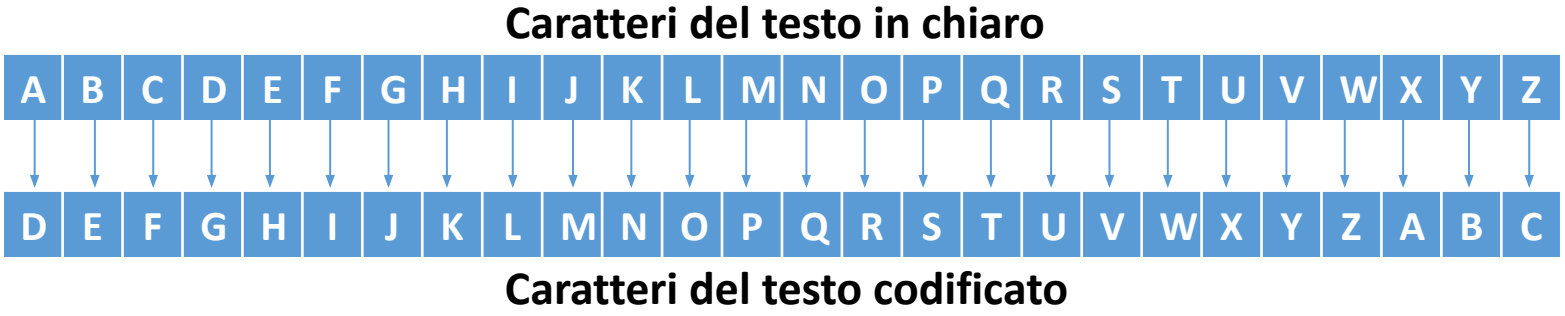
Ad esempio, con PARAM=3, la lettera E viene sostituita dalla lettera H, J dalla M, W dalla Z, **Y** dalla **B**, e così via.



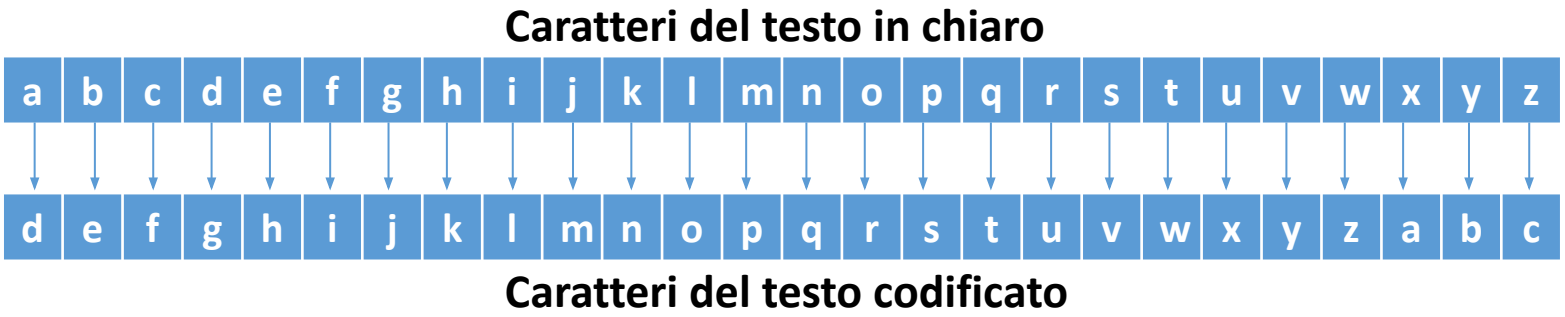

**N.B.: se procedendo "in avanti" si arriva oltre la fine dell'alfabeto, si riparte dall'inizio (scorrendo il numero di posti rimanenti)**

# Codifica di Cesare

Ricapitolando, con PARAM=3, lo schema di codifica è quello riportato in figura:




Analogamente, ci sarà uno schema di codifica per le lettere minuscole:




- Dato PARAM=26 (o multiplo), il testo in chiaro deve coincidere con il testo codificato.
- Dato PARAM=*x* + 26*k* (*x* e *k* interi positivi), la schema di codifica è lo stesso del caso in cui PARAM=*x.*

# Esercizio

Scrivere la funzione che implementi la codifica di Cesare di un singolo carattere.

- La funzione deve codificare solo caratteri lettera (maiuscole e minuscole).
- Le lettere maiuscole vanno codificate con lettere maiuscole; analoga procedura per le lettere minuscole.
- Caratteri non lettera (cifre, punteggiatura, ecc.) vanno lasciati "in chiaro".

# Idea di soluzione

- La funzione deve codificare solo caratteri lettera (maiuscole e minuscole).
- Le lettere maiuscole vanno codificate con lettere maiuscole; analoga procedura per le lettere minuscole.
- Caratteri non lettera (cifre, punteggiatura, ecc.) vanno lasciati «in chiaro».

Dobbiamo definire la funzione **codificaCesare**, che riceve in input il carattere da codificare *c* e il numero PARAM di "passi in avanti" nell'alfabeto, e restituisce il carattere codificato *c'*.

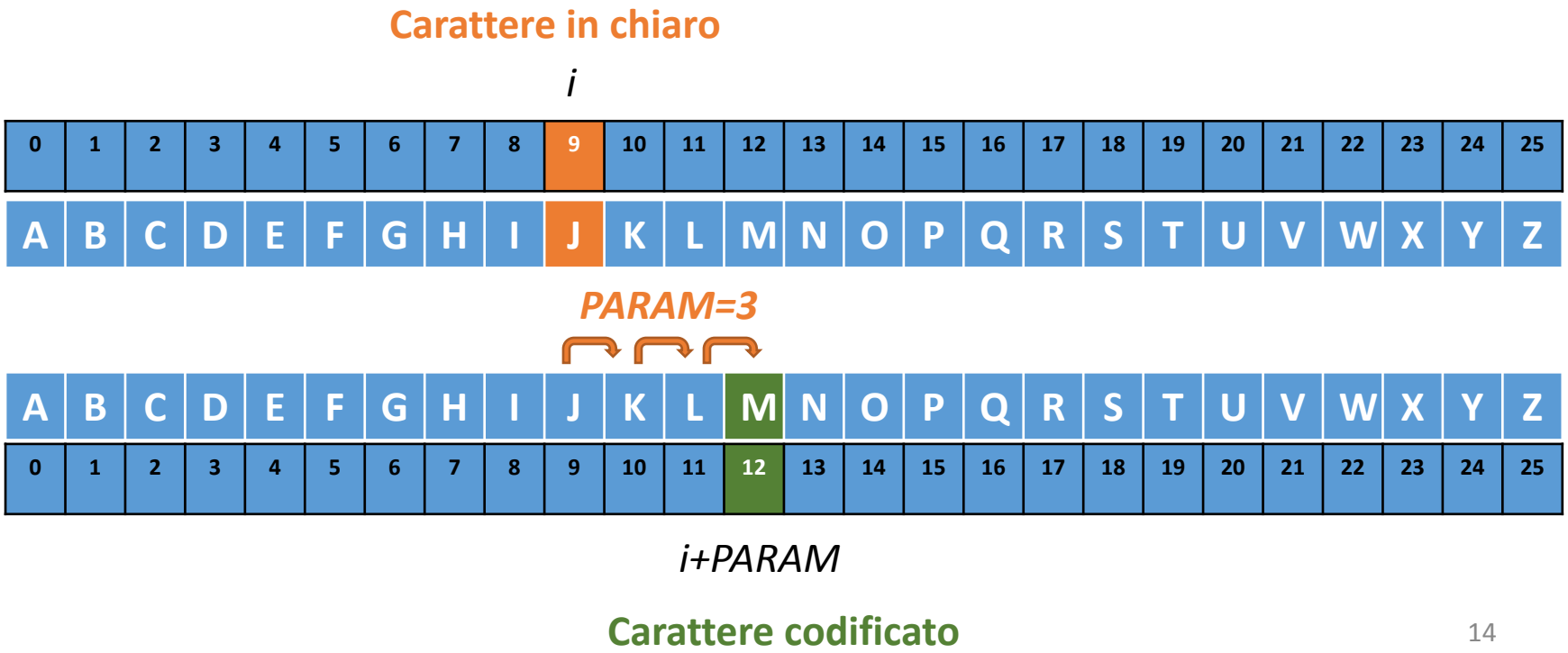
Quindi:

- Se *c* è un carattere lettera
  1. Se *c* è un carattere maiuscolo, si applica la codifica di Cesare per le lettere maiuscole
  2. Altrimenti, si applica la codifica di Cesare per le lettere minuscole
- Altrimenti, *c'*=*c*

Analizziamo per primo il caso 1 (codifica lettere maiuscole), sfruttando la posizione del carattere *c* nell'alfabeto…

# Codifica di Cesare per maiuscole

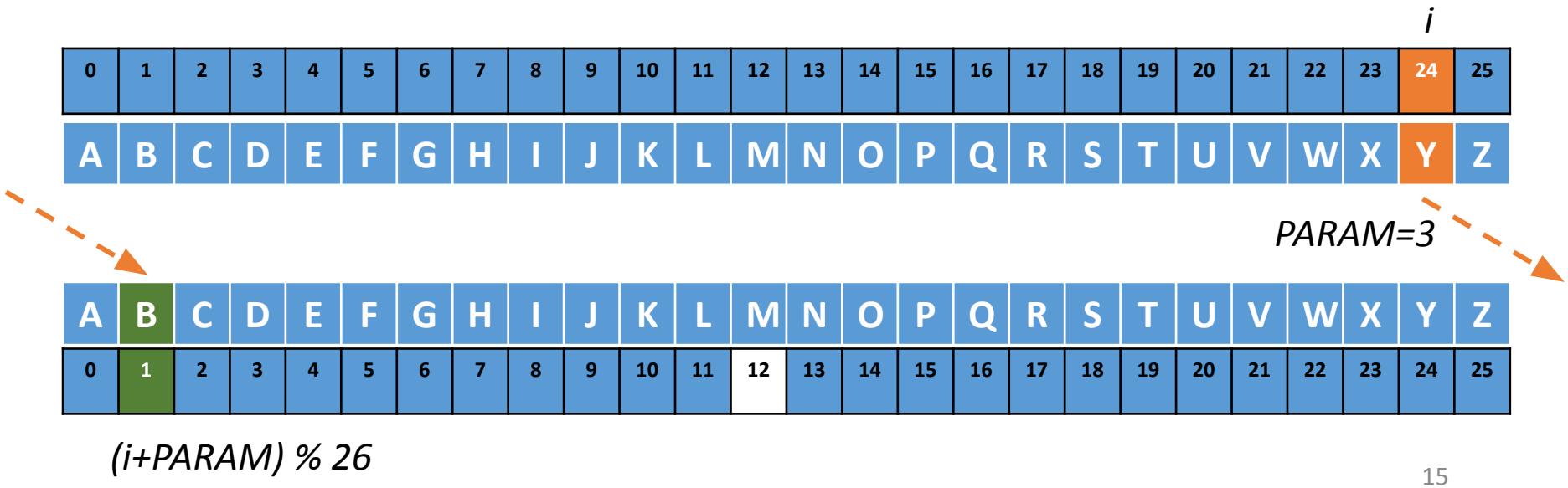
Definito *i* come indice del carattere *c* nell'alfabeto, il carattere *c'* sarà in posizione *i+PARAM*.



# Codifica di Cesare per maiuscole

Definito *i* come indice del carattere *c* nell'alfabeto, il carattere *c'* sarà in posizione *i+PARAM.*

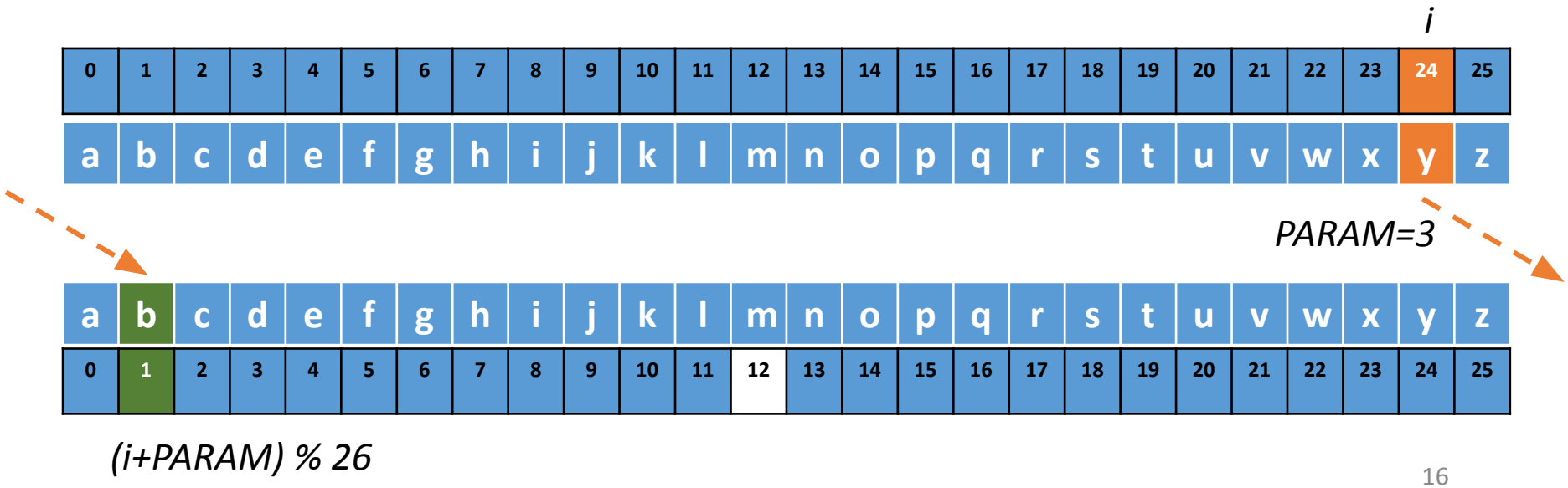
Devo però gestire il caso in cui *i+PARAM* individua una posizione oltre l'ultimo carattere dell'alfabeto (*i+PARAM>25).* E' sufficiente ripartire dal primo carattere, muovendosi del numero di "passi" rimanenti, quindi *c'* sarà in posizione (*i+PARAM) % 26.*



# Codifica di Cesare per minuscole

Analogamente alla codifica per le lettere maiuscole, *c'* sarà in posizione *(i+PARAM) % 26.*

Quello che cambia è che i caratteri nell'alfabeto sono lettere minuscole.



# Codice Python

```
def codificaCesare(c, param):
  cp = ''
  maiuscole = ['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']
  minuscole = ['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']
  if c.isalpha():
    if c.isupper():
      indice_c = maiuscole.index(c)
      indice_cp = (indice_c + param) % 26
      cp = maiuscole[indice_cp]
    else:
      indice_c = minuscole.index(c)
      indice_cp = (indice_c + param) % 26
      cp = minuscole[indice_cp]
  else:
    cp = c
  return cp
```

# Codice Python

Definizione delle liste per contenere i caratteri maiuscoli e minuscoli, **in ordine alfabetico**

```
def codificaCesare(c, param):
  cp = ''
  maiuscole = ['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']
  minuscole = ['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']
  if c.isalpha():
    if c.isupper():
      indice_c = maiuscole.index(c)
      indice_cp = (indice_c + param) % 26
      cp = maiuscole[indice_cp]
    else:
      indice_c = minuscole.index(c)
      indice_cp = (indice_c + param) % 26
      cp = minuscole[indice_cp]
  else:
    cp = c
  return cp
```

# Codice Python

```
def codificaCesare(c, param):
  cp = ''
  maiuscole = ['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']
  minuscole = ['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']
  if c.isalpha():
    if c.isupper():
      indice_c = maiuscole.index(c)
      indice_cp = (indice_c + param) % 26
      cp = maiuscole[indice_cp]
    else:
      indice_c = minuscole.index(c)
      indice_cp = (indice_c + param) % 26
      cp = minuscole[indice_cp]
  else:
    cp = c
  return cp
```

Se il carattere ***c*** è una lettera allora esegui la codifica …

Altrimenti non eseguire la codifica di ***c***

# Codice Python

```
def codificaCesare(c, param):
  cp = ''
  maiuscole = ['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']
  minuscole = ['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']
  if c.isalpha():
    if c.isupper():
      indice_c = maiuscole.index(c)
      indice_cp = (indice_c + param) % 26
      cp = maiuscole[indice_cp]
    else:
      indice_c = minuscole.index(c)
      indice_cp = (indice_c + param) % 26
      cp = minuscole[indice_cp]
  else:
    cp = c
  return cp
```

Se ***c*** è una lettera **maiuscola**

Calcolo l'indice ***i***

Calcolo l'indice del carattere ***c'***

Identifico il carattere ***c'***

# Codice Python

```
def codificaCesare(c, param):
  cp = ''
  maiuscole = ['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']
  minuscole = ['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']
  if c.isalpha():
    if c.isupper():
      indice_c = maiuscole.index(c)
      indice_cp = (indice_c + param) % 26
      cp = maiuscole[indice_cp]
    else:
      indice_c = minuscole.index(c)
      indice_cp = (indice_c + param) % 26
      cp = minuscole[indice_cp]
  else:
    cp = c
  return cp
```

Se ***c*** è una lettera **minuscola**

Calcolo l'indice ***i***

Calcolo l'indice del carattere ***c'***

Identifico il carattere ***c'***

# Versione 1

```
def codificaCesare(c, param):
  cp = ''
  maiuscole = ['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']
  minuscole = ['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']
  if c.isalpha():
    if c.isupper():
      indice_c = maiuscole.index(c)
      indice_cp = (indice_c+param)% 26
      cp = maiuscole[indice_cp]
    else:
      indice_c = minuscole.index(c)
      indice_cp = (indice_c+param)% 26
      cp = minuscole[indice_cp]
  else:
    cp = c
  return cp
```

# Versione 2

```
def codificaCesare(c, param):
  cp = ''
  isUp = True
  alfabeto = ['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']
  if c.isalpha():
    if c.islower():
      isUp = False
      c = c.upper()
    indice_c = alfabeto.index(c)
    indice_cp = (indice_c+param)% 26
    cp = alfabeto[indice_cp]
    if not isUp:
      cp = cp.lower()
  else:
    cp = c
  return cp
```

# Versione 1

```
def codificaCesare(c, param):
  cp = ''
  maiuscole = ['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']
  minuscole = ['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']
  if c.isalpha():
    if c.isupper():
      indice_c = maiuscole.index(c)
      indice_cp = (indice_c+param)% 26
      cp = maiuscole[indice_cp]
    else:
      indice_c = minuscole.index(c)
      indice_cp = (indice_c+param)% 26
      cp = minuscole[indice_cp]
  else:
    cp = c
  return cp
```

# Versione 2

```
def codificaCesare(c, param):
  cp = ''
  isUp = True
  alfabeto = ['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']
  if c.isalpha():
    if c.islower():
      isUp = False
      c = c.upper()
    indice_c = alfabeto.index(c)
    indice_cp = (indice_c+param)% 26
    cp = alfabeto[indice_cp]
    if not isUp:
      cp = cp.lower()
  else:
    cp = c
  return cp
```

# Versione 2

```
def codificaCesare(c, param):
  cp = ''
  isUp = True
  alfabeto = ['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']
  if c.isalpha():
    if c.islower():
      isUp = False
      c = c.upper()
    indice_c = alfabeto.index(c)
    indice_cp = (indice_c+param)% 26
    cp = alfabeto[indice_cp]
    if not isUp:
      cp = cp.lower()
  else:
    cp = c
  return cp
```

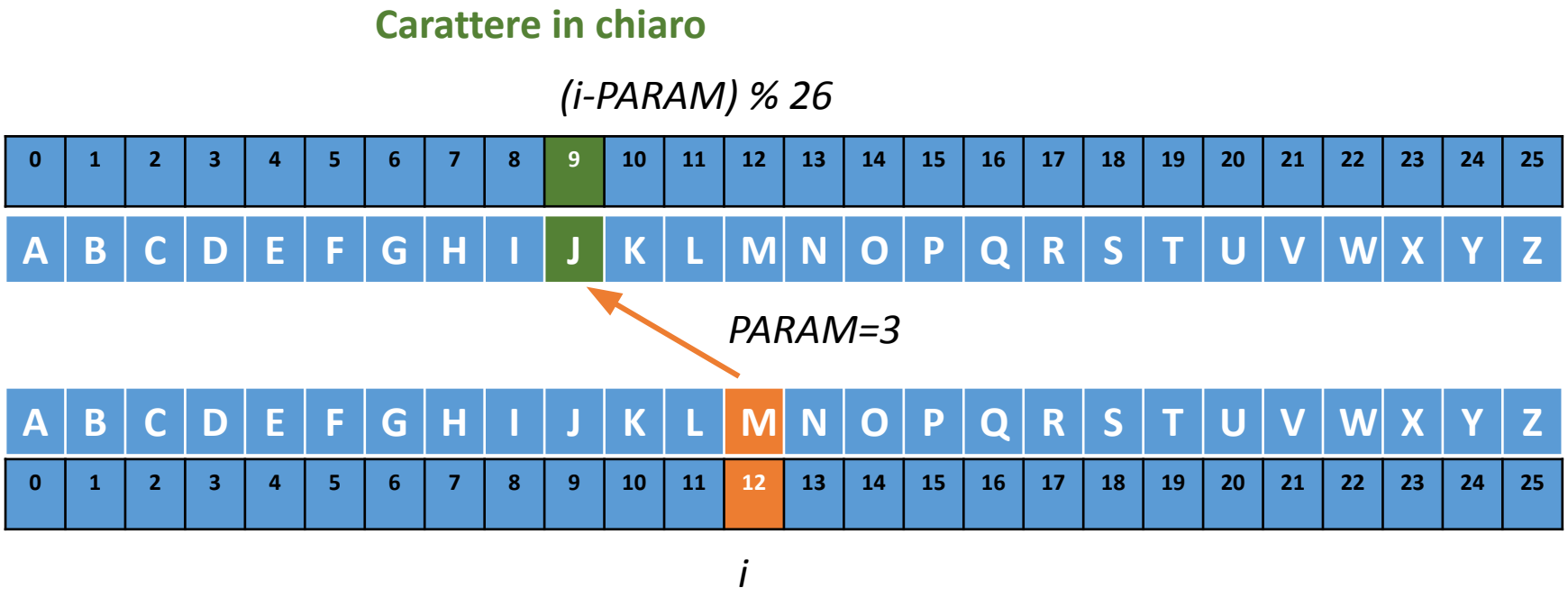
# Versione 3

```
def indiceCarattere(c):
  if c.islower():
    return ord(c)-ord('a')
  else:
    return ord(c)-ord('A')

def codificaCesare(c, param):
  cp = ''
  isUp = True
  if c.isalpha():
    if c.islower():
      isUp = False
      c = c.upper()
    indice_c = indiceCarattere(c)
    indice_cp = (indice_c+param) % 26
    cp = chr(ord('A') + indice_cp)
    if not isUp:
      cp = cp.lower()
  else:
    cp = c
  return cp
```

# Decodifica di Cesare

Definito *i* come indice del carattere *c'* nell'alfabeto, il carattere *c* sarà in posizione (*i-PARAM) % 26.*



# Codice Python - Decodifica

```
def indiceCarattere(c):
  if c.islower():
    return ord(c)-ord('a')
  else:
    return ord(c)-ord('A')

def decodificaCesare(c, param):
  cp = ''
  isUp = True
  if c.isalpha():
    if c.islower():
      isUp = False
      c = c.upper()
    indice_c = indiceCarattere(c)
    indice_cp = (indice_c - param) % 26
    cp = chr(ord('A') + indice_cp)
    if not isUp:
      cp = cp.lower()
  else:
    cp = c
  return cp
```

# Codice Python - Decodifica

```
def indiceCarattere(c):
  if c.islower():
    return ord(c)-ord('a')
  else:
    return ord(c)-ord('A')

def decodificaCesare(c, param):
  cp = ''
  isUp = True
  if c.isalpha():
    if c.islower():
      isUp = False
      c = c.upper()
    indice_c = indiceCarattere(c)
    indice_cp = (indice_c - param) % 26
    cp = chr(ord('A') + indice_cp)
    if not isUp:
      cp = cp.lower()
  else:
    cp = c
  return cp
```

# Codice Python – Codifica e decodifica

```
def codDecodCesare(c, param, codifica):
  cp = ''
  isUp = True
  if c.isalpha():
    if not codifica:
      param = -param
    if c.islower():
      isUp = False
      c = c.upper()
    indice_c = indiceCarattere(c)
    indice_cp = (indice_c + param) % 26
    cp = chr(ord('A') + indice_cp)
    if not isUp:
      cp = cp.lower()
  else:
    cp = c
  return cp
```

# Codice Python – Codifica e decodifica

```
def codDecodCesare(c, param, codifica):
  cp = ''
  isUp = True
  if c.isalpha():
    if not codifica:
      param = -param
    if c.islower():
      isUp = False
      c = c.upper()
    indice_c = indiceCarattere(c)
    indice_cp = (indice_c + param) % 26
    cp = chr(ord('A') + indice_cp)
    if not isUp:
      cp = cp.lower()
  else:
    cp = c
  return cp
```

Parametro codifica:

- Se **True**, eseguire la **codifica**
- Se **False**, eseguira la **decodifica**

Se decodifica, il numero di passi è negativo (spostamento indietro)

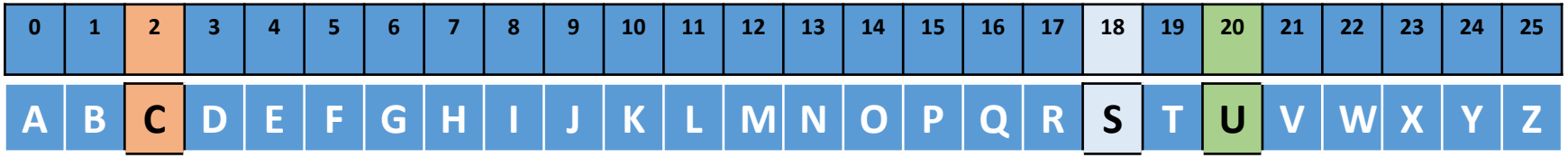
# Codifica di Vigenère

La cifratura di Vigenère cifra ciascuna lettera del testo originario usando un'opportuna cifratura di Cesare, diversa ogni volta.

La specifica cifratura di Cesare da utilizzare per ciascuna lettera dipende dalla PASSWORD.

| | | | | |
|---|---|---|---|---|
| C | I | A | O | Testo in chiaro |
| S | O | L | E | PASSWORD |
| U | W | L | S | Testo codificato |

Il carattere codificato si ottiene spostando il carattere in chiaro di un numero fisso di passi, pari all'indice del carattere corrispondente nella PASSWORD




# Codifica di Vigenère

| | | | |
|---|---|---|---|
| C | I | A | O |
| S | O | L | E |
| U | W | L | S |

Testo in chiaro

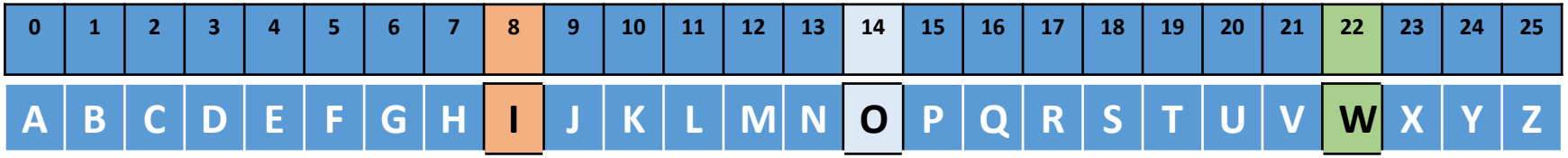
PASSWORD

Testo codificato

*Definiamo la funzione ind(carattere), che restituisce l'indice del carattere nell'alfabeto*

L'indice nell'alfabeto dell'i-esimo carattere del testo codificato è la somma degli indici dell'i-esimo carattere in chiaro e dell' i-esimo carattere della PASSWORD.

1. *ind('U')=ind('C')+ind('S')*




# Codifica di Vigenère

| | | | |
|---|---|---|---|
| C | I | A | O |
| S | O | L | E |
| U | W | L | S |

Testo in chiaro

PASSWORD

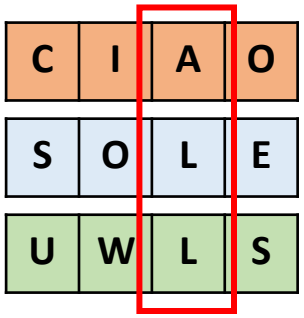
Testo codificato

L'indice nell'alfabeto dell'i-esimo carattere del testo codificato è la somma degli indici dell'i-esimo carattere in chiaro e dell' i-esimo carattere della PASSWORD.

1. *ind('U')=ind('C')+ind('S')*
2. *ind('W')=ind('I')+ind('O')*

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 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 |

# Codifica di Vigenère



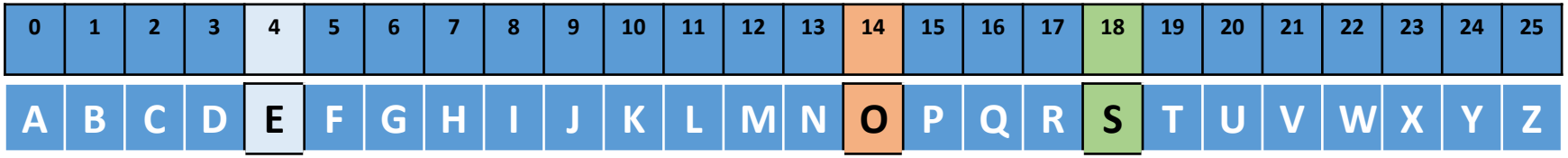

Testo in chiaro

PASSWORD

Testo codificato

L'indice nell'alfabeto dell'i-esimo carattere del testo codificato è la somma degli indici dell'i-esimo carattere in chiaro e dell' i-esimo carattere della PASSWORD.

1. *ind('U')=ind('C')+ind('S')*
2. *ind('W')=ind('I')+ind('O')*
3. *ind('L')=ind('A')+ind('L')*




# Codifica di Vigenère

| | | | |
|---|---|---|---|
| C | I | A | O |
| S | O | L | E |
| U | W | L | S |

Testo in chiaro

PASSWORD

Testo codificato

L'indice nell'alfabeto dell'i-esimo carattere del testo codificato è la somma degli indici dell'i-esimo carattere in chiaro e dell' i-esimo carattere della PASSWORD.

1. *ind('U')=ind('C')+ind('S')*
2. *ind('W')=ind('I')+ind('O')*
3. *ind('L')=ind('A')+ind('L')*
4. *ind('S')=ind('O')+ind('E')*

# Codifica di Vigenère

| | | | |
|---|---|---|---|
| C | I | A | O |
| S | O | L | E |
| U | W | L | S |

Testo in chiaro

PASSWORD

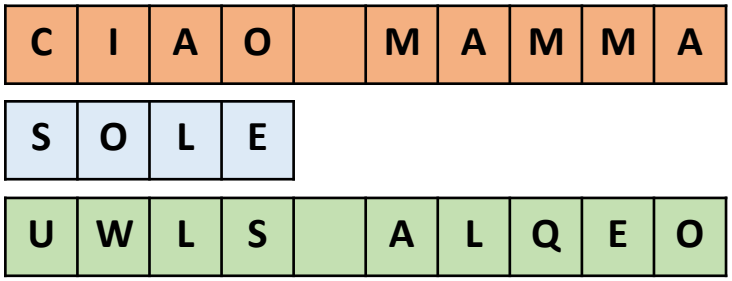
Testo codificato

L'indice nell'alfabeto dell'i-esimo carattere del testo codificato è la somma degli indici dell'i-esimo carattere in chiaro e dell' i-esimo carattere della PASSWORD.

1. *ind('U') = ind('C') + ind('S')*
2. *ind('W') = ind('I') + ind('O')*
3. *ind('L') = ind('A') + ind('L')*
4. *ind('S') = ind('O') + ind('E')*

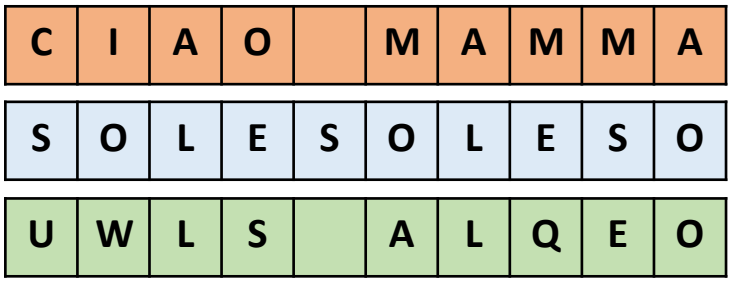
Notate come gli indici dei caratteri della PASSWORD agiscano come valori di PARAM per la cifratura di Cesare

# Codifica di Vigenère




Nel caso in cui la PASSWORD sia più corta del testo in chiaro, si riprende dalla prima lettera della PASSWORD.

E' possibile immaginare il procedimento di codifica come se lo scenario fosse quello sotto:



# Codice Python

```
# Gestione dell'input secondo specifiche dell'esercizio
…

testoChiaro = … # è il testo in chiaro da codificare
password = … # è la password per la codifica
codifica = … # può essere True o False (codifica o decodifica)

for i in range(len(testoChiaro)):
  indice_psw = i % len(password)
  param = indiceCarattere(password[indice_psw])
  c = testoChiaro[i]
  cp = codDecodCesare(c, param, codifica)
  print(cp, end='')
```

# Codice Python

```
# Gestione dell'input secondo specifiche dell'esercizio
…

testoChiaro = … # è il testo in chiaro da codificare
password = … # è la password per la codifica
codifica = … # può essere True o False (codifica o decodifica)

for i in range(len(testoChiaro)):
  indice_psw = i % len(password)
  param = indiceCarattere(password[indice_psw])
  c = testoChiaro[i]
  cp = codDecodCesare(c, param, codifica)
  print(cp, end='')
```

Indice carattere corrente della password

# Codice Python

```
# Gestione dell'input secondo specifiche dell'esercizio
…

testoChiaro = … # è il testo in chiaro da codificare
password = … # è la password per la codifica
codifica = … # può essere True o False (codifica o decodifica)

for i in range(len(testoChiaro)):
  indice_psw = i % len(password)
  param = indiceCarattere(password[indice_psw])
  c = testoChiaro[i]
  cp = codDecodCesare(c, param, codifica)
  print(cp, end='')
```

Indice nell'alfabeto del carattere corrente della password

# Codice Python

```
# Gestione dell'input secondo specifiche dell'esercizio
…

testoChiaro = … # è il testo in chiaro da codificare
password = … # è la password per la codifica
codifica = … # può essere True o False (codifica o decodifica)

for i in range(len(testoChiaro)):
  indice_psw = i % len(password)
  param = indiceCarattere(password[indice_psw])
  c = testoChiaro[i]
  cp = codDecodCesare(c, param, codifica)
  print(cp, end='')
```

Carattere corrente del testo in chiaro

# Codice Python

```
# Gestione dell'input secondo specifiche dell'esercizio
…

testoChiaro = … # è il testo in chiaro da codificare
password = … # è la password per la codifica
codifica = … # può essere True o False (codifica o decodifica)

for i in range(len(testoChiaro)):
  indice_psw = i % len(password)
  param = indiceCarattere(password[indice_psw])
  c = testoChiaro[i]
  cp = codDecodCesare(c, param, codifica)
  print(cp, end='')
```

Codifica/decodifica con codice di Cesare