

# Laboratorio 5 — Cifrario Byte-a-Byte con Password Mnemonica

*Live Demo — byte\_cipher.c*

Università di Catania — Dipartimento di Matematica e Informatica

**Corso di Laurea in Informatica**

Programmazione 1 e Laboratorio (Canale O-Z)

---

## Descrizione del Programma

Si scriva un programma in linguaggio C, chiamato `byte_cipher.c`, che implementi un cifrario simmetrico su tutto lo spazio dei byte (0-255) basato su una password mnemonica fornita dall'utente.

Il programma calcola un seed deterministico a partire dalla password mnemonica di tipo stringa, genera una permutazione pseudo-casuale dei 256 possibili valori di un `unsigned char` (da 0 a 255) tramite il generatore `rand()` inizializzato con tale seed, e costruisce la corrispondente mappa di decodifica inversa. Infine, legge una riga di testo e applica la codifica o la decodifica byte-a-byte.

Il programma deve gestire correttamente i flussi di input/output e preservare il terminatore nullo `'\0'` in modo che il testo risultante sia stampabile come stringa standard del C.

## Requisiti di Implementazione

Il programma deve essere strutturato nei seguenti punti:

**PUNTO A: Acquisizione Modalità:** Leggere da standard input un carattere che indichi la modalità (C per codificare, D per decodificare).

- Si stampi il prompt: `Scegli modalita' (C = codifica, D = decodifica):` .
- Se il carattere inserito è diverso da C, c, D, d, il programma deve stampare `Errore: modalita' non valida.` e richiedere nuovamente l'inserimento in un ciclo indefinito finché non viene inserita una modalità corretta.

**PUNTO B: Acquisizione Password e Testo:**

- Stampare il prompt **Inserisci la password** (max 255 car.): e acquisire la password mnemonica (stringa) tramite `fgets`. Rimuovere l'eventuale carattere newline finale `'\n'`. Se la password risulta vuota, il programma deve stampare **Errore: password vuota.** e richiederla nuovamente in un ciclo indefinito.
- Stampare il prompt **Inserisci il testo** (max 999 car.): e acquisire il testo da cifrare o decifrare tramite `fgets`, rimuovendo il newline finale.

### PUNTO C: Derivazione del Seed e delle Mappe:

- Definire ed invocare la funzione `unsigned int derive_seed(const char *password)` che scorre la password ed accumula i caratteri partendo da un valore iniziale `seed = 0`. Per ogni carattere `c = password[i]` fino al terminatore, aggiornare il seed secondo la formula:

$$\text{seed}_{\text{nuovo}} = \text{seed}_{\text{precedente}} \cdot 31 + (\text{unsigned char})c$$

- Definire ed invocare la funzione `void build_map(unsigned char map[256], unsigned int seed)` per inizializzare un array lineare da 0 a 255 ed applicarvi lo shuffle di Fisher-Yates tramite la funzione `rand()` opportunamente inizializzata con `srand(seed)`.
- Definire ed invocare la funzione `void build_inverse_map(const unsigned char map[256], unsigned char inv_map[256])` per calcolare la permutazione inversa, in modo tale che `inv_map[map[i]] == i`.

### PUNTO D: Cifratura / Decifratura e Stampa:

- Definire le funzioni di cifratura/decifratura per scorrere la stringa e applicare la sostituzione dei caratteri byte-a-byte preservando il terminatore `'\0'`.
- A seconda della modalità, stampare a schermo il testo modificato anticipato dall'opportuna etichetta: **Testo codificato:** `[testo]` oppure **Testo decificato:** `[testo]`.
- Il programma deve terminare con `exit` status 0.

# Output di Controllo (Esempi di Esecuzione)

## Caso 1 — Cifratura di una frase completa

Inserendo i dati:

```
C
itvf
cifrario simmetrico
```

Il programma deve stampare a schermo:

```
Scegli modalita' (C = codifica, D = decodifica): Inserisci la password (max 255 car
↪ .): Inserisci il testo (max 999 car.): Testo codificato: [P]FEFPh_?PVV=HFP[
↪ h
```

*(Exit code atteso: 0)*

## Caso 2 — Decifratura di una frase cifrata

Inserendo i dati:

```
D
itvf
[P]FEFPh_?PVV=HFP[h
```

Il programma deve stampare a schermo:

```
Scegli modalita' (C = codifica, D = decodifica): Inserisci la password (max 255 car
↪ .): Inserisci il testo (max 999 car.): Testo decodificato: cifrario
↪ simmetrico
```

*(Exit code atteso: 0)*