Esercizio Programmazione a.a. 2025/2026

Programmazione Procedurale con Laboratorio, Corso di Laurea in Informatica, UniPG

Canali per le domande: ricevimento, email: francesco.santini@unipg.it, Telegram: @safran

1 Cifrario di Vernam (versione con operatori bitwise in C)

Il *cifrario di Vernam* è un sistema di cifratura simmetrica basato sull'operazione bitwise `XOR` (`^`) tra i bit del testo in chiaro e quelli di una chiave di uguale lunghezza. È considerato una forma pratica del cosiddetto *one-time pad*, cioè un sistema in cui la chiave viene usata una sola volta e deve essere lunga almeno quanto il messaggio da cifrare. (Fonte: Wikipedia)

In un'implementazione in C, sia il testo in chiaro sia la chiave sono trattati come sequenze di caratteri, cioè array di tipo `char`. Ogni carattere può essere interpretato come un numero intero tra 0 e 255, corrispondente al suo codice ASCII.

La cifratura avviene applicando l'operatore bitwise `XOR` (di seguito indicato come $\oplus$, o `^` in C) tra ciascun carattere del messaggio P_i (plaintext) e il corrispondente carattere della chiave K_i:

$$C_i = P_i \oplus K_i$$

dove $\oplus$ rappresenta l'operazione di `XOR` bit-a-bit. Il risultato C_i è un nuovo carattere che costituisce il testo cifrato.

Nel codice C, la formula può essere realizzata come:

```
cipher[i] = plaintext[i] ^ key[i];
```

Per decifrare il messaggio, basta applicare nuovamente la stessa operazione `XOR` con la medesima chiave, poiché vale:

$$(P_i \oplus K_i) \oplus K_i = P_i$$

ovvero la `XOR` è un'operazione *auto-inversa*.

Se la chiave è realmente casuale e usata una sola volta, il sistema è *teoricamente inviolabile*.

Alcune osservazioni pratiche:

- La chiave deve avere la stessa lunghezza del messaggio; in alternativa può essere ripetuta ciclicamente.
- L'operatore `^` agisce direttamente sui bit dei caratteri `char`, permettendo di cifrare qualsiasi simbolo testuale o binario.

Ulteriori informazioni teoriche e storiche sono disponibili online.[1].

[1] https://it.wikipedia.org/wiki/Cifrario_di_Vernam

2 Esercizio

Scrivere un programma su un file singolo di nome *soluzione.c* (non modificare il nome del file) che per prima cosa prende in input da tastiera una stringa con caratteri senza vincoli, contenente quindi anche numeri, simboli di punteggiatura, etc (è in gergo il nostro *plaintext*); fare riferimento alla Figura 1 per la dimensione massima della stringa e come leggerla: *fgets*[2] è una funzione della libreria *stdio.h*. La stringa da cifrare deve essere lunga al massimo 24 caratteri.

Come prima cosa, si elimini dalla stringa ricevuta tutti i caratteri che non fanno parte dell'alfabeto, cioè a-z, A-Z. Supporre di utilizzare l'alfabeto inglese (26 lettere). Inoltre, le lettere maiuscole diventano minuscole. Per esempio, la stringa "Ciao, come stai?" deve essere trasformata in "ciaocomestai".

Se l'utente richiede il cifrario di Vernam, si deve richiedere in input da tastiera anche la parola chiave k, e poi restituire come output la stringa cifrata (stampare ciascun elemento della stringa come intero invece che come carattere, dato che la cifratura potrebbe portare ad ottenere caratteri di controllo che non sono facilmente stampabili). In gergo, il risultato della cifratura è chiamato *cyphertext*. Per la parola chiave, supporre di leggere una stringa di lunghezza massimo 8; fare sempre riferimento a Figura 1 su come leggere una stringa con lunghezza massima stabilita. Subito sotto il messaggio cifrato, si deve stampare anche il risultato della decifrazione del cyphertext utilizzando la stessa chiave k e utilizzando sempre la stessa operazione $\oplus$, in modo da poter verificare che il risultato sia uguale al plaintext originale (in questo caso, stampare ciascun elemento della decifratura come carattere).

In Tabella 1 è riportato un esempio di come viene cifrato il plaintext "ciao" con k uguale a "neon".

Pos	Plain char	ASCII dec	ASCII bin	Key char	Key dec	Key bin	XOR dec	XOR bin	XOR hex
1	c	99	01100011	n	110	01101110	13	00001101	0x0D
2	i	105	01101001	e	101	01100101	12	00001100	0x0C
3	a	97	01100001	o	111	01101111	14	00001110	0x0E
4	o	111	01101111	n	110	01101110	1	00000001	0x01

Tabella 1. Cifratura Vernam del testo `ciao` con la chiave `neon` (codifica ASCII).

Dopo questa operazione, si deve chiedere di nuovo all'utente se vuole eseguire una nuova cifratura, cifrando un nuovo messaggio con una nuova chiave, oppure terminare il programma.

Suggerimento. Possono essere utilizzate (solo) le librerie di sistema, come la libreria *string.h*[3] e la libreria di sistema *ctype.h*[4] ed in particolare la funzione *islower()* e *isalpha()* per controllare se un carattere è minuscolo oppure una lettera. La funzione

[2] `https://www.tutorialspoint.com/c_standard_library/c_function_fgets.htm`
[3] `https://www.tutorialspoint.com/c_standard_library/string_h.htm`.
[4] `https://www.tutorialspoint.com/c_standard_library/ctype_h.htm`.

```
char stringa[24];
fgets(stringa, sizeof(stringa), stdin);
```

Figura 1. Esempio per definire la stringa da leggere e leggere la stringa da tastiera con *fgets*.

isalpha() in *ctype.h* è in particolare utile per controllare se un carattere è una lettere oppure no.

Attenzione 1 *Attenzione, in C si possono verificare errori tipici durante la lettura dal buffer di input. Per esempio un carattere di nuova linea* \n *può rimanere nel buffer dopo una prima operazione di lettura, ed una seguente operazione di lettura può leggere quello invece di attendere input da tastiera da parte dell'utente. Oppure, se ciò che viene letto da tastiera è più lungo del limite utilizzato nella fgets, i caratteri in più possono rimanere nel buffer di lettura. Utilizzare* stackoverflow[5,6]*, dove si può trovare tutta l'informazione ed esempi di cui si ha bisogno. Riassumendo, il buffer di lettura deve essere pulito tra due letture consecutive.*

Attenzione 2 *Non utilizzare la funzione* fflush()[7] *perché sullo standard input, associato alla tastiera, (*stdin*) il suo comportamento è indefinito in C*[8]

3 Sottomissione

Si accettano solo sottomissioni attraverso GitHub all'indirizzo `https://classroom.github.com/a/zFElCNIU`. È necessario creare un account GitHub prima di cliccare sul link. Lo studente deve essere in grado di verificare da solo se il *push* delle modifiche effettuate in locale abbia avuto successo o meno (non si risponde a domande come "Può controllare se ho sottomesso correttamente?"). L'utilizzo delle funzionalità minime di Github fa infatti parte delle conoscenze richiesta dal corso.

Per quanto riguarda l'esercizio si prega di **non** creare *branch* secondari, ma di modificare il solamente template nel branch principale: al momento del download per la correzione, i branch secondari **non** sono visibili e quindi il repository risulterebbe vuoto. Infine, non è necessario avere i diritti di amministratore da parte dello studente, che infatti non sono concessi in modo da per esempio non poter rendere il repository pubblico (in quanto esercizio individuale).

Nel template dell'esercizio è presente un file *.gitignore* che previene il *push* di file eseguibili (*.exe* o *.out*) e altri, che **non** devono far parte dal versionamento offerto da GitHub. L'esercizio deve essere contenuto nel file *soluzione.c*, che è già presente nel template della cartella GitHub quando viene clonata in locale.

---

[5] `https://stackoverflow.com/questions/2907062/fgets-instructions-gets-skipped-why`.
[6] `https://stackoverflow.com/questions/7898215/how-to-clear-input-buffer-in-c`.
[7] `https://www.tutorialspoint.com/c_standard_library/c_function_fflush.htm`.
[8] `https://stackoverflow.com/questions/2979209/using-fflushstdin`.

La scadenza per la sottomissione è **Domenica 23 Novembre** (ore 23:59). Nel file README.md del repository aggiungere, ricordarsi di aggiungere 1) nome, 2) cognome e 3) matricola.

4 Note finali

Compilare aggiungendo i flag *-std=c11 -Wall* (per esempio `gcc crypto.c -std=c11 -Wall`), per essere sicuri di seguire lo standard *C 2011* e farsi segnalare tutti i *warning* rispettivamente. I *warning* devono essere **tutti** rimossi.

Possono essere utilizzare solamente le librerie standard fornite da C (per esempio *stdio.h*, *string.h*, *ctype.h*, etc.).