Esercizio Programmazione a.a. 2024/2025

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 Vigenère

In crittografia, il cifrario di Vigenère è il più semplice dei cifrari polialfabetici. Si basa sull'uso di un versetto per controllare l'alternanza degli alfabeti di sostituzione, concetto introdotto per la prima volta da Giovan Battista Bellaso ne *La cifra del Sig. Giovan Battista Belaso* del 1553.[1]

Il metodo si può considerare una generalizzazione del cifrario di Cesare[2]; invece di spostare sempre dello stesso numero di posti la lettera da cifrare, questa viene spostata di un numero di posti variabile ma ripetuto, determinato in base ad una parola chiave, da concordarsi tra mittente e destinatario, e da scrivere ripetutamente sotto il messaggio, carattere per carattere; la chiave era detta anche verme, per il motivo che, essendo in genere molto più corta del messaggio, deve essere ripetuta molte volte sotto questo. Fare riferimento all'esempio presente nella pagina Wikipedia dell cifrario di Vigenère, riportato in Figura 1.

```
Testo in chiaro  - RAPPORTOIMMEDIATO
Verme            - VERMEVERMEVERMEVE
Testo cifrato    - MEGBSMXFUQHIUUEOS
```

Figura 1. Cifrario di Vigenère con parola chiave "verme" [Fonte Wikipedia].

2 Cifrario Playfair

Il cifrario Playfair, o quadrato di Playfair, è una tecnica di cifratura simmetrica manuale basata su un cifrario monoalfabetico a due lettere. Lo schema fu inventato nel 1854 dal fisico inglese Sir Charles Wheatstone ma prende il nome del suo amico Lord Playfair Barone di St. Andrews, che cercò di divulgarne l'uso.[3] Il cifrario Playfair si basa sull'uso di una matrice 5×5 contenente una parola chiave. La memorizzazione della chiave e 4 semplici regole sono tutto ciò che è richiesto per creare la tabella 5 per 5 e usare il codice. Per capire regole e funzionamento, documentarsi su Internet[4]

Come chiave di cifratura, utilizzare la parola chiave "esempioplayfair" come nella pagina Wikipedia, e in Figura 2.

[1] `https://it.wikipedia.org/wiki/Cifrario_di_Vigenère`.
[2] `https://it.wikipedia.org/wiki/Cifrario_di_Cesare`.
[3] `https://it.wikipedia.org/wiki/Cifrario_Playfair`.
[4] `https://www.geeksforgeeks.org/playfair-cipher-with-examples/`.

Figura 2. Cifrario Playfair [Fonte Wikipedia].

3 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 3 per la dimensione massima della stringa e come leggerla: *fgets* 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".

A questo punto, l'utente sceglie con quale dei due metodi sopra descritti vuole cifrare la stringa:

1. Il cifrario di Vigenère;
2. Il cifrario Playfair.

Se l'utente richiede il cifrario di Vigenère, si deve richiedere in input da tastiera anche la parola chiave *k*, e poi restituire come output la stringa cifrata. 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 3 su come leggere una stringa con lunghezza massima stabilita.

Se l'utente richiede il cifrario di Playfair, utilizzare come chiave fissa la matrice in Figura 2. Anche in questo caso, restituire in output la stringa cifrata.

Dopo una delle due operazioni, si deve tornare a chiedere di nuovo all'utente se vuole eseguire una nuova cifratura, cioè le operazioni 1) o 2).

Suggerimento. Possono essere utilizzate (solo) le librerie di sistema, come la libreria *string.h*[5] e la libreria di sistema *ctype.h*[6] ed in particolare la funzione *islower()* e *isalpha()* per controllare se un carattere è minuscolo oppure una lettera. La funzione *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*

[5] `https://www.tutorialspoint.com/c_standard_library/string_h.htm`.
[6] `https://www.tutorialspoint.com/c_standard_library/ctype_h.htm`.

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

Figura 3. Esempio per definire la stringa da leggere e leggere la stringa da tastiera con *fgets*.

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[7,8]*, 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()[9] *perché sullo standard input, associato alla tastiera, (*stdin*) il suo comportamento è indefinito in C*[10]

4 Sottomissione

Si accettano solo sottomissioni attraverso GitHub all'indirizzo `https://classroom.github.com/a/HZNd_2r1`. 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.

La scadenza per la sottomissione è **Lunedi 25 Novembre** (ore 23:59). Nel file README.md del repository aggiungere, ricordarsi di aggiungere 1) nome, 2) cognome e 3) matricola.

---

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

5 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.).