CORSO DI PROGRAMMAZIONE A-L
A.A. 2017-18

Dispensa 17

Laboratorio

Dott. Filippo Piccinini
e-mail: f.piccinini@unibo.it

Ringrazio sinceramente il Prof. Mirko Ravaioli per avere gentilmente condiviso il materiale usato negli anni precedenti.

Esempio: Gioco dell'Impiccato

Il gioco

L'impiccato è un gioco per due giocatori. Uno dei giocatori sceglie segretamente una frase; l'altro deve indovinarla entro un certo numero di tentativi. La frase viene mostrata al secondo giocatore mascherata: mostrando al posto delle vocali il simbolo '+' e al posto delle consonanti il simbolo '-', gli altri caratteri (spazi o punteggiatura) rimangono invariati. Il gioco termina quando la frase viene indovinata, o si finiscono i tentativi a disposizione.

L'algoritmo risolutivo per il gioco potrebbe essere:

1. Chiedere al primo giocatore di inserire la frase da indovinare
2. Mascherare la frase inserita, sostituendo al posto delle vocali il simbolo '+' e al posto delle consonanti il simbolo '-'
3. Chiedere al secondo giocatore di inserire una lettera
4. Verificare che la lettera inserita sia presente all'interno della frase, se la lettera è presente sostituire il simbolo con la lettera indovinata
5. Stampare la frase mascherata con le eventuali lettere indovinate
6. Se la frase non è stata indovinata vai al punto 3, altrimenti comunica all'utente che ha indovinato la frase
7. Esci dal gioco

L'algoritmo sopra descritto non risolve a pieno il gioco in quanto non tiene traccia del numero massimo di tentativi che un giocatore può fare, in particolare quando si supera il numero di tentativi bisognerebbe interrompere il gioco in quanto l'utente ha perso. Per tenere traccia del numero di tentativi si potrebbe utilizzare un contatore incrementato ogni volta che l'utente inserisce una nuova lettera. L'algoritmo potrebbe diventare quindi:

1. Chiedere al primo giocatore di inserire la frase da indovinare
2. Mascherare la frase inserita, sostituendo al posto delle vocali il simbolo '+' e al posto delle consonanti il simbolo '-'
3. Chiedere al secondo giocatore di inserire una lettera
4. Incrementare il contatore di un'unità
5. Verificare che la lettera inserita sia presente all'interno della frase, se la lettera è presente sostituire il simbolo con la lettera indovinata
6. Stampare la frase mascherata con le eventuali lettere indovinate
7. Se la frase è stata indovinata comunicare la vittoria e andare al punto 10
8. Se il numero di tentativi (e quindi il valore del contatore) è uguale al massimo dei tentativi disponibili comunica all'utente che ha perso e vai al punto 10
9. Se la frase non è stata indovinata vai al punto 3
10. Esci dal gioco

Questo secondo algoritmo però permette di indovinare la frase inserendo esclusivamente lettera per lettera, potrebbe capitare che il numero di lettere diverse in una frase sia superiore al numero di tentativi e quindi il gioco non consentirebbe all'utente di arrivare in fondo e quindi di vincere. La cosa migliore da fare potrebbe essere quella di dare la possibilità all'utente di indovinare la frase ad un certo punto del gioco o meglio di

poter scegliere se inserire una nuova lettera o indovinare la frase. In questo modo è possibile assegnare anche un punteggio sulla base dei tentativi fatti. Il nuovo algoritmo potrebbe essere:

1. Chiedere al primo giocatore di inserire la frase da indovinare
2. Mascherare la frase inserita, sostituendo al posto delle vocali il simbolo '+' e al posto delle consonanti il simbolo '-'
3. Chiedere al secondo giocatore se vuole indovinare la frase o inserire una lettera
4. Se il giocatore vuole indovinare la frase vai al punto 5 altrimenti al punto 7
5. Chiedere all'utente di inserire la frase
6. Se la frase è stata indovinata comunicare la vittoria e andare al punto 14 altrimenti vai al punto 3
7. Chiedere all'utente di inserire una lettera
8. Incrementare il contatore delle lettere inserite di un'unità
9. Verificare che la lettera inserita sia presente all'interno della frase, se la lettera è presente sostituire il simbolo con la lettera indovinata
10. Stampare la frase mascherata con le eventuali lettere indovinate
11. Se le lettere sono state tutte scoperte comunicare la vittoria e andare al punto 14
12. Se il numero di tentativi (e quindi il valore del contatore) è uguale al massimo dei tentativi disponibili comunica all'utente che ha perso e vai al punto 14
13. Se la frase non è stata indovinata vai al punto 3
14. Esci dal gioco

Questo algoritmo però consentirebbe ad un utente di provare all'infinito di indovinare la frase, senza mai inserire una lettera, potremo quindi tenere traccia anche del numero di tentativi di indovinare la frase tutto in un colpo. L'algoritmo finale potrebbe essere:

1. Chiedere al primo giocatore di inserire la frase da indovinare
2. Mascherare la frase inserita, sostituendo al posto delle vocali il simbolo '+' e al posto delle consonanti il simbolo '-'
3. Chiedere al secondo giocatore se vuole indovinare la frase o inserire una lettera
4. Se il giocatore vuole indovinare la frase vai al punto 5 altrimenti al punto 8
5. Chiedere all'utente di inserire la frase
6. Se la frase è stata indovinata comunicare la vittoria e andare al punto 15 altrimenti vai al punto 7
7. Dato che la frase non è stata indovinata incrementare il contatore che tiene traccia dei tentativi, se tale numero supera il massimo consentito comunicare all'utente che ha perso e andare al punto 15 altrimenti andare al punto 3
8. Chiedere all'utente di inserire una lettera
9. Incrementare il contatore delle lettere inserite di un unità
10. Verificare che la lettera inserita sia presente all'interno della frase, se la lettera è presente sostituire il simbolo con la lettera indovinata
11. Stampare la frase mascherata con le eventuali lettere indovinate
12. Se le lettere sono state tutte scoperte comunicare la vittoria e andare al punto 15
13. Se il numero di tentativi (e quindi il valore del contatore) è uguale al massimo dei tentativi disponibili comunica all'utente che ha perso e vai al punto 15
14. Se la frase non è stata indovinata vai al punto 3

15. Esci dal gioco

Sicuramente potrebbero essere fatte molte altre migliorie, però per implementare il gioco di base questo ultimo algoritmo dovrebbe rispondere a tutte le esigenze.

Implementazione l'algoritmo

Suddivisione in sottoproblemi

L'ultimo algoritmo potrebbe essere scomposto in 3 sotto problemi:

1. Richiesta e lettura della frase da indovinare
2. Sostituzione delle vocali con il carattere '+' e delle consonanti con il carattere '-'
3. Ricerca di una lettera all'interno di una frase

Richiesta e lettura della frase da indovinare

Il linguaggio C non ha un tipo di dato primitivo per la gestione delle stringhe, infatti esse vengono considerate come una sequenza di caratteri con l'aggiunta di un carattere speciale di terminazione \0 per individuarne la fine. Quindi per esempio volendo memorizzare la frase "ciao mondo" in un array di 15 caratteri il risultato che otterremo sarà:

c	i	a	o		m	o	n	d	o	\0				
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14

Ipotizzando che la frase da indovinare non superi i 20 caratteri (incluso spazi, caratteri di punteggiatura...) per la memorizzazione della stringa dovremo dichiarare un array di caratteri di dimensione 21: 20 per memorizzare la stringa e 1 per il carattere terminatore:

```
char frase[21];
```

L'iterazione con l'utente è molto importante e quindi bisogna comunicare attraverso delle stampe a video le istruzioni che dovranno essere eseguite, quindi attraverso una funzione *printf()*:

```
printf("Inserire la frase da indovinare e premere INVIO: ");
```

La lettura della frase inserita dall'utente potrebbe essere fatta attraverso la funzione *scanf()* utilizzando il carattere di conversione %s apposito per le stringhe:

```
scanf("%s",frase);
```

Da notare che all'interno della chiamata alla funzione *scanf()* la variabile "frase" non viene preceduta dall'operatore d'indirizzo & (e-commerciale) in quanto tale operatore va utilizzato esclusivamente per le variabili ti tipo primitivo.

```
#include <stdio.h>

int main()
{
	char frase[21];
	printf("Inserire la frase da indovinare e premere INVIO: ");
	scanf("%s",frase);

	return 0;
}
```

Questa soluzione non è ottimale in quanto utilizzando il carattere di conversione %s la funzione *scanf()* considera come fine della frase NON solo la pressione del tasto INVIO ma anche il carattere spazio. Quindi inserendo la frase: "ciao mondo", il programma memorizzerà all'interno dell'array "frase" solamente la parola "ciao". Per ovviare a questo problema potremo utilizzare il carattere di conversione %[^...] il quale ci consente di specificare quali caratteri non sono ammessi nella lettura e quindi quelli che verranno considerati come fine stringa (al posto dei puntini vanno inseriti i caratteri non ammessi). In questo caso l'unico carattere non ammesso è l'INVIO (o meglio "nuova linea" o ritorno a carrello) individuato dal carattere speciale \n:

```
	scanf("%[^\n]",frase);
```

La funzione *scanf()* non è l'unica funzione che consente la lettura da tastiera, infatti esistono altre funzioni per l'input di riga, ad esempio *gets()*. Tale funzione si limita a leggere una riga da stdin e la memorizza in una stringa (array di caratteri). La funzione legge caratteri finché non incontra un carattere di nuova riga (\n) o la fine della riga:

```
	gets(frase);
```

In questo modo tutti i caratteri inseriti, spazi inclusi verranno letti e memorizzati all'interno dell'array frase.

Potemmo ottimizzare il nostro programma inserendo una costante per gestire la dimensione per la stringa dato che tale dimensione sicuramente dovrà essere utilizzata in altri punti del nostro codice. Quindi il sorgente potrebbe essere:

```
#include <stdio.h>

#define LUNGHEZZA_FRASE 20

int main()
{
	char frase[LUNGHEZZA_FRASE+1];
	printf("Inserire la frase da indovinare e premere INVIO: ");
	gets(frase);

	return 0;
}
```

Potrebbe capitare che un utente inserisca una frase con un numero di caratteri maggiore di 20, per evitare tale problema potremo stampare a video il limite massimo consentito ma questo non ci garantirebbe al 100% che l'inserimento da parte dell'utente risulti corretto. Sia la funzione *scanf()* che la funzione *gets()* entrano in azione SOLO dopo la pressione del tasto INVIO, accedendo al buffer di lettura fino a trovare il carattere terminatore. Per ovviare al problema potremo utilizzare la funzione *getchar()* la quale ci consente di leggere un carattere alla volta dal buffer.

La funzione *getchar()* attende la ricezione di un carattere da stdin. Poichè *getchar()* è una funzione di input con buffer, non viene ricevuto alcun carattere finché l'utente non preme INVIO. Tuttavia ogni tasto premuto viene immediatamente mostrato sullo schermo. Quando si preme INVIO, tutti i caratteri immessi, incluso

quello di nuova riga vengono inviati a stdin dal sistema operativo. La funzione *getchar()* restituisce i caratteri uno alla volta:

```
i = 0;
while ((ch = getchar()) !=  13 && i < LUNGHEZZA_FRASE)
      frase[i++] = ch;

frase[i] = '\0';
```

I caratteri di volta in volta letti vengono inseriti all'interno dell'array fino ad arrivare al carattere terminatore o alla dimensione massima dell'array. Notare che il carattere terminatore è in questo caso segnalato dal 13, che nella tabella ASCII corrisponde a "carriage return" (in Italiano ritorno a capo). In questo modo riusciamo ad evitare che all'interno dell'array "frase" che ci siano più caratteri di quelli consentiti ma non ad impedire che l'utente scriva più di 20 caratteri. Per evitare questo ultimo problema potremo utilizzare le funzioni *getch()* e *getche()* le quali consentono di leggere direttamente il carattere successivo dal flusso stdin. In poche parole man mano che l'utente inserisce i caratteri questi vengono letti dalla funzione. Utilizziamo la funzione *getche()* in quanto funzione con eco cioè mostra i caratteri letti nel flusso stdout:

```
i = 0;
while ((ch = getche()) != 13 && i < LUNGHEZZA_FRASE)
      frase[i++] = ch;

frase[i] = '\0';
```

L'utente potrebbe inserire una frase vuota cioè quindi premere direttamente INVIO, ovviamente questa cosa non dovrebbe essere consentita. Potremo risolvere anche quest'ultima problematica controllando il valore della variabile "i" dopo il ciclo while: se la variabile vale 0 allora non sono stati inseriti caratteri utili:

```
i = 0;
do
{
      while ((ch=getche()) != 13 && i < LUNGHEZZA_FRASE)
            frase[i++] = ch;
      if (i == 0)
            printf("Hai inserito una frase vuota! \n");
}
while(i == 0);

frase[i] = '\0';
```

Il tutto viene inserito all'interno di un ciclo do-while per ripetere l'operazione fino a quando l'utente non inserirà un valore corretto, e quindi i diventi maggiore di 0. Per fare in modo che l'utente possa decidere ad un certo punto dell'inserimento della frase di uscire dal programma potremo considerare per esempio che alla pressione del tasto - (meno) il programma termini. Per ottenere questo effetto dovremo controllare di volta in volta il carattere inserito dall'utente:

```
i = 0;
do
{
      while ((ch=getche()) != '\n' && ch != '\r' && i < LUNGHEZZA_FRASE)
      {
            if (ch == '-')
                  exit(0);
            frase[i++] = ch;
      }
      if (i == 0)
```

```
                printf("Hai inserito una frase vuota! \n");
        }
        while(i == 0);

        frase[i] = '\0';
```

Notare che in questo caso il termine della frase è stato controllato con i comandi ch != '\n' && ch != '\r', che comunque agiscono allo stesso modo di ch != 13. Per essere più chiari: nella tabella ASCII in valore 13 decimale indica il "Carriage Return", che precisamente sarebbe il \r. tuttavia è opportuno controllare anche per il \n che indica "New Line".

Il programma per l'inserimento della frase diventa quindi:

```
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>

#define LUNGHEZZA_FRASE 20

int main()
{
        char frase[LUNGHEZZA_FRASE+1];
        int i;
        char ch;

        printf("Inserire la frase da indovinare e premere INVIO [- esci]: ");
        i = 0;
        do
        {
                while ((ch=getche()) != 13 && i < LUNGHEZZA_FRASE)
                {
                        if (ch == '-')
                                exit(0);
                        frase[i++] = ch;
                }
                if (i == 0)
                        printf("Hai inserito una frase vuota! \n");
        }
        while(i == 0);

        frase[i] = '\0';

        return 0;
}
```

Sostituzione delle vocali e delle consonanti

Per poter sostituire tutte le vocali con il carattere '+' e tutte le consonanti con il carattere '-', dovremo accedere singolarmente ad ogni cella dell'array contenente la stringa per valutarne ogni singolo carattere:

```
        for (i = 0; i < LUNGHEZZA_FRASE; i++)
        {
                if (frase[i] == 'a' || frase[i] == 'e' || frase[i] == 'i' ||
                        frase[i] == 'o' || frase[i] == 'u')
                        frase[i] = '+';
                else
                        frase[i] = '-';
        }
```

Invece di utilizzare un'istruzione if per i controlli potremo utilizzare un'istruzione switch:

```
for (i = 0; i < LUNGHEZZA_FRASE; i++)
{
      switch(frase[i])
      {
      case 'a':
      case 'e':
      case 'i':
      case 'o':
      case 'u':
            frase[i] = '+';
            break;
      default:
            frase[i] = '-';
}
```

Le soluzioni presentate in precedenza non sono ottimali in quanto vengono considerate indipendentemente dalla lunghezza della frase tutte le celle presenti all'interno dell'array.

Per ovviare al problema potremo inserire un controllo ulteriore all'interno della condizione del for per valutare che la cella considerata non contenga il carattere terminatore della stringa:

```
for (i = 0; i < LUNGHEZZA_FRASE && frase[i] != '\0'; i++)
{
      if (frase[i] == 'a' || frase[i] == 'e' || frase[i] == 'i' ||
            frase[i] == 'o' || frase[i] == 'u')
            frase[i] = '+';
      else
            frase[i] = '-';
}
```

Se l'utente inserisce lettere maiuscole, dato che il linguaggio C è case sensitive e quindi il carattere 'a' minuscolo risulterà diverso dal carattere 'A' maiuscolo, il programma potrebbe rispondere in maniera errata. Potremo quindi all'interno dell'if o dello switch considerare tutti i caratteri (in questo caso le vocali) anche in maiuscolo oppure utilizzando la funzione *tolower()* potremo trasformare per la verifica tutti i caratteri in minuscolo:

```
for (i = 0; i < LUNGHEZZA_FRASE && frase[i] != '\0'; i++)
{
      carattere = tolower(frase[i]);
      if (carattere == 'a' || carattere == 'e' || carattere == 'i' ||
            carattere == 'o' || carattere == 'u')
            frase[i] = '+';
      else
            frase[i] = '-';
}
```

La nuova soluzione presenta comunque ancora degli errori in termini di algoritmo: inserendo una frase con spazi o caratteri speciali questi verranno sostituiti con il simbolo '-' in quanto l'unico controllo è quello che valuta se il carattere corrisponde ad una vocale.

Le soluzioni a questo secondo problema possono essere varie: potremo inserire altri controlli all'interno dell'istruzione else per valutare se il carattere è una costante oppure potremo creare all'interno dello switch tanti case uno per ogni costante (allo stesso modo che è stato fatto per le vocali).

Dato che i caratteri vengono memorizzati attraverso dei numeri descritti attraverso la tabella ASCII, e dato che le lettere dell'alfabeto (in minuscolo) all'interno di tale tabella hanno codici che variano da 97 a 122 il programma potrebbe essere ottimizzato nel seguente modo:

```
for (i = 0; i < LUNGHEZZA_FRASE && frase[i] != '\0'; i++)
{
      carattere = tolower(frase[i]);
      if (carattere >= 97 && carattere <= 122)
      {
            if (carattere == 'a' || carattere == 'e' ||
                  carattere == 'i' || carattere == 'o' ||
                  carattere == 'u')
                  frase[i] = '+';
            else
                  frase[i] = '-';
      }
}
```

In questo modo verranno sostituite solamente le lettere appartenenti all'alfabeto. Rimangono però escluse tutte le lettere accentate. Per gestire anche questa problematica dovremmo utilizzare variabili di tipo "unsigned char" in quanto tali caratteri fanno parte della tabella ASCII estesa. Comunque per evitare di complicare troppo il programma non considereremo le lettere accentate.

Ricerca di una lettera all'interno di una frase

Per trovare tutte le occorrenze di una lettera all'interno di una frase bisogna eseguire dei confronti singolarmente con tutti i caratteri della stringa data.

Considerando quanto detto nei paragrafi precedenti, la stringa inserita dall'utente viene modificata sostituendo le varie lettere dell'alfabeto con dei caratteri speciali, quindi non sarà più possibile per il programma conoscere quali lettere effettivamente saranno presenti all'interno della frase dopo le varie modifiche a meno che non esita una copia della frase iniziale o la gestione della stringa crittografata, con i simboli '+' e '-', non venga fatta con un altro array:

```
for (i = 0; i < LUNGHEZZA_FRASE && frase[i] != '\0'; i++)
{
      carattere = tolower(frase[i]);
      if (carattere >= 97 && carattere <= 122)
      {
            if (carattere == 'a' || carattere == 'e' ||
                  carattere == 'i' || carattere == 'o' ||
                  carattere == 'u')
                  fraseCrittografata[i] = '+';
            else
                  fraseCrittografata [i] = '-';
      }
      else
            fraseCrittografata[i] = frase[i];
}
fraseCrittografata[i] = '\0';
```

La variabile "fraseCrittografata" è un array della stessa dimensione e tipologia dell'array "frase". Ora con questa modifica, presa una lettera da cercare dall'utente potremo individuare tutte le occorrenze e sostituire ogni volta nella frase crittografata la lettera indovinata:

```
printf("Iserire una lettera e premere INVIO: ");
scanf("%c",&lettera);
for (i = 0; i < LUNGHEZZA_FRASE && frase[i] != '\0'; i++)
{
      if (tolower(frase[i]) == tolower(lettera))
            fraseCrittografata[i] = frase[i];

}
```

Da notare che nella funzione *scanf()* la variabile "lettera" viene preceduta dall'operatore indirizzo e-commerciale (&) e che all'interno del for i confronti delle lettere vengono fatti considerando le lettere in minuscolo sempre per evitare il problema della gestione case sensitive.

Per evitare che l'utente debba premere INVIO dopo ogni inserimento potremo utilizzare la funzione *getche()* descritta in precedenza. Inoltre dato che possono essere inseriti caratteri speciali diversi dalle lettere dell'alfabeto potremo eseguire un controllo per verificare la correttezza dell'inserimento:

```
do
{
      printf("Iserire una lettera: ");
      fflush(stdin);
      lettera = tolower(getche());
      if (lettera < 97 || lettera > 122)
            printf("Errore di inserimento: carattere non valido\n");
}
while(lettera < 97 || lettera > 122);

for (i = 0; i < LUNGHEZZA_FRASE && frase[i] != '\0'; i++)
{
      if (tolower(frase[i]) == lettera)
            fraseCrittografata[i] = frase[i];

}
```

Per poter comunicare all'utente se la lettera inserita è presente o meno all'interno della frase nascosta bisogna valutare al di fuori del corpo del for se si è entrati almeno una volta all'interno del if che confronta la lettera inserita con l'i-esima della frase. Questo tipo di controllo potrebbe essere fatto con una variabile impostata ad un certo valore prima del for e modificata esclusivamente all'interno del corpo dell'istruzione if. Se si accede al corpo dell'if, e quindi se esiste almeno un'occorrenza della lettera inserita dall'utente all'interno della frase, allora la variabile verrà modificata altrimenti rimarrà con il suo valore iniziale:

```
trovato = 0;
for (i = 0; i < LUNGHEZZA_FRASE && frase[i] != '\0'; i++)
{
      if (tolower(frase[i]) == lettera)
      {
            fraseCrittografata[i] = frase[i];
            trovato = 1;
      }
}

if (trovato)
      printf("Lettera trovata!");
```

```
else
      printf("La lettera non è presente nella frase!");
```

Ottimizzazione dei sottoproblemi in relazione al programma

Dato che bisogna dare la possibilità ad un utente di indovinare la frase oltre che inserire una lettera o eventualmente di uscire dal gioco, potremo associare queste funzionalità alla pressione di tasti speciali quindi considerare che se premendo il tasto '-' (meno) si esce dal gioco, premendo il tasto '?' è possibile dare la soluzione, altrimenti viene considerata la lettera inserita e fatta una ricerca all'interno della frase:

```
do
{
      printf("Iserire una lettera [- esci; ? indovina]: ");
      fflush(stdin);
      lettera = tolower(getche());
      if (lettera == '-')
            exit(0);
      if (lettera != '?' && (lettera < 97 || lettera > 122))
            printf("Errore di inserimento: carattere non valido\n");
}
while(lettera != '?' && (lettera < 97 || lettera > 122));

if (lettera == '?')
{
      printf("Inserisci la frase nascosta e premere INVIO: ");

      i = 0;
      while ((ch =getche()) != 13 && i < LUNGHEZZA_FRASE)
            fraseUtente[i++] = ch;
      fraseUtente[i] = '\0';
      if (strcmp(strlwr(frase), strlwr(fraseUtente)) == 0)
      {
            printf("Hai indovinato!");
            exit(0);
      }
      else
            printf("Frase non corretta!");
}
else
{
      /*parte dedicata alla ricerca della lettera*/
      ....
}
```

La funzione *strlwr()* consente di trasformare una stringa tutta con caratteri in minuscolo, mentre la funzione *strcmp()* consente di confrontare due stringhe.

L'inserimento della frase da parte dell'utente viene gestito come all'inizio del programma per inserire la frase da indovinare, per gli stessi motivi descritti in precedenza.

Volendo limitare il numero di tentativi per indovinare la frase e un numero massimo di lettere da poter inserire dovranno essere utilizzare due contatori rispettivamente per il numero di lettere inserite e il numero di tentativi di indovinare. Tali contatori verranno di volta in volta confrontati con i massimi consentiti (tutta la parte del gioco viene inserita all'interno di un ciclo per ripetere gli inserimenti fino a quando l'utente non raggiunge i limiti dei tentativi o indovina la frase o decide di uscire dal gioco):

```
numeroLettere = 0;
numeroTentativi = 0;

do
{
	do
	{
		printf("Iserire una lettera [- esci; ? indovina]: ");
		fflush(stdin);
		lettera = tolower(getche());
		if (lettera == '-')
			exit(0);
		if (lettera != '?' && (lettera < 97 || lettera > 122))
			printf("Errore di inserimento: carattere non valido\n");
	}
	while(lettera != '?' && (lettera < 97 || lettera > 122));

	if (lettera == '?')
	{
		printf("Inserisci la frase nascosta e premere INVIO: ");
		while ((ch =getche()) != 13 && i < LUNGHEZZA_FRASE)
			fraseUtente[i++] = ch;
		fraseUtente[i] = '\0';

		numeroTentativi++;
		if (strcmp(strlwr(frase), strlwr(fraseUtente)) == 0)
		{
			printf("Hai indovinato!");
			exit(0);
		}
		else
			printf("Frase non corretta!");

		if (numeroTentativi >= MASSIMO_TENTATIVI)
		{
			printf("Hai esaurito i tentativi a tua disposizione!");
			exit(0);
		}
	}
	else
	{
		numeroLettere++;

		trovato = 0;
		for (i = 0; i < LUNGHEZZA_FRASE && frase[i] != '\0'; i++)
		{
			if (tolower(frase[i]) == lettera)
			{
				fraseCrittografata[i] = frase[i];
				trovato = 1;
			}
		}

		if (trovato)
			printf("Lettera trovata!");
		else
			printf("La lettera non è presente nella frase!");
		printf("\n%s", fraseCrittografata);
```

```
            if (numeroLettere >= MASSIMO_INSERIMENTI)
            {
                  printf("Hai raggiunto il numero massimo di inserimenti!");
                  printf("Prova a indovinare la frase nascosta: ");
                  i = 0;
                  while ((ch=getche())!='\n' && ch!='\r' &&  i<LUNGHEZZA_FRASE)
                        fraseUtente[i++] = ch;

                  if (strcmp(strlwr(frase), strlwr(fraseUtente)) == 0)
                        printf("Hai indovinato!");
                  else
                        printf("Frase non corretta! Hai perso!");
                  exit(0);
            }
      }
}
while(1);
```

Volendo fare le cose a modo, bisognerebbe che il numero massimo di inserimenti possibili risulti uguale al massimo al numero di lettere differenti presenti all'interno della frase:

```
      int elencoLettereAlfabeto[26];
      int numeroMassimoInserimentiLettere;

      .../*altro codice*/

      for (i = 0; i < 26; i++)
            elencoLettereAlfabeto[i] = 0;

      numeroMassimoInserimentiLettere = 0;
      for (i = 0; i < LUNGHEZZA_FRASE && frase[i] != '\0'; i++)
      {
            lettera = tolower(frase[i]);
            if (lettera >= 97 && lettera <= 122)
            {
                  if (elencoLettereAlfabeto[lettera-97] != 1)
                        numeroMassimoInserimentiLettere++;
                  elencoLettereAlfabeto[lettera-97] = 1;
            }
      }
```

dove "elencoLettereAlfabeto" è un array di interi, ogni cella associata logicamente ad una lettera dell'alfabeto. Tale array verrà utilizzato per tenere traccia delle lettere della frase già considerate nel conteggio.

Unione dei sottoproblemi per ottenere il programma

Sicuramente potrebbero essere fatte molte altre migliorie al programma, comunque quanto detto risulta più che sufficiente per creare una buona soluzione:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <conio.h>
#include <ctype.h>

#define LUNGHEZZA_FRASE 20
```

```
#define MASSIMO_TENTATIVI 3

int main()
{
	char fraseCrittografata[LUNGHEZZA_FRASE+1];
	char frase[LUNGHEZZA_FRASE+1];
	char fraseUtente[LUNGHEZZA_FRASE+1];
	int i, trovato;
	char carattere, lettera, ch;
	int numeroMassimoInserimentiLettere;
	int numeroLettere, numeroTentativi;
	int elencoLettereAlfabeto[26];

	printf("\nInserire la frase da indovinare e premere INVIO [- esci]:\n");
	i = 0;
	do
	{
		fflush(stdin); /* Per pulire stdin. */

		while ((ch=getche()) != 13 && i < LUNGHEZZA_FRASE)
		{
			if (ch == '-')
			{
				exit(0);
				system("pause");
		}
				frase[i++] = ch;
		}
		if (i == 0)
			printf("\nHai inserito una frase vuota!\n");
	}
	while(i == 0);
	frase[i] = '\0';


	for (i = 0; i < LUNGHEZZA_FRASE && frase[i] != '\0'; i++)
	{
		carattere = tolower(frase[i]);
		if (carattere >= 97 && carattere <= 122)
		{
			if (carattere == 'a' || carattere == 'e' ||
				carattere == 'i' || carattere == 'o' ||
				carattere == 'u')
				fraseCrittografata[i] = '+';
			else
				fraseCrittografata [i] = '-';
		}
		else
			fraseCrittografata[i] = frase[i];
	}
	fraseCrittografata[i] = '\0';


	for (i = 0; i < 26; i++)
		elencoLettereAlfabeto[i] = 0;

	numeroMassimoInserimentiLettere = 0;
	for (i = 0; i < LUNGHEZZA_FRASE && frase[i] != '\0'; i++)
	{
		lettera = tolower(frase[i]);
		if (lettera >= 97 && lettera <= 122)
		{
			if (elencoLettereAlfabeto[lettera-97] != 1)
			{
				numeroMassimoInserimentiLettere++;
			}
			elencoLettereAlfabeto[lettera-97] = 1;
```

```
		}
	}

	numeroLettere = 0;
	numeroTentativi = 0;

	system("cls"); /* Per pulire lo schermo. */
	do
	{
		do
		{

			printf("\n\nFrase da indovinare: \n%s\n\n", fraseCrittografata);
			printf("Iserire una lettera [- esci; ? indovina]: ");
			fflush(stdin);
			lettera = tolower(getche());
			if (lettera == '-')
			{
				exit(0);
				system("pause");
		}
			if (lettera != '?' && (lettera < 97 || lettera > 122))
				printf("\nErrore di inserimento: carattere non valido\n");
		}
		while(lettera != '?' && (lettera < 97 || lettera > 122));

		if (lettera == '?')
		{
			printf("\nInserisci la frase nascosta e premere INVIO: ");
			i = 0;
			while ((ch =getche()) != 13 && i < LUNGHEZZA_FRASE)
				fraseUtente[i++] = ch;
		fraseUtente[i] = '\0';
			numeroTentativi++;
			if (strcmp(strlwr(frase), strlwr(fraseUtente)) == 0)
			{
				printf("\nHAI INDOVINATO!\n");
				system("pause");
				exit(0);
			}
			else
				printf("\nFrase non corretta!\n");

			if (numeroTentativi >= MASSIMO_TENTATIVI)
			{
				printf("\nHai esaurito i tentativi a tua disposizione!\n");
				system("pause");
				exit(0);
			}
		}
		else
		{
			numeroLettere++;

			trovato = 0;
			for (i = 0; i < LUNGHEZZA_FRASE && frase[i] != '\0'; i++)
			{
				if (tolower(frase[i]) == lettera)
				{
					fraseCrittografata[i] = frase[i];
					trovato = 1;
				}
			}

			if (trovato)
				printf("\nLettera trovata!\n");
			else
```

```
                        printf("\nLa lettera non è presente nella frase!\n");

                    if (numeroLettere >= numeroMassimoInserimentiLettere)
                    {
                        printf("\nRaggiunto il numero massimo di inserimenti!\n");
                        printf("\nProva a indovinare la frase nascosta:\n");
                    i = 0;
                        while ((ch=getche())!=13 && i <LUNGHEZZA_FRASE)
                            fraseUtente[i++] = ch;

                        fraseUtente[i] = '\0';
                        if (strcmp(strlwr(frase), strlwr(fraseUtente)) == 0)
                            printf("\nHAI INDOVINATO!\n");
                        else
                            printf("\nFrase non corretta! Hai perso!\n");
                        system("pause");
                        exit(0);
                    }
            }
        }
        while(1);

    return 0;
}
```

Modificare il codice sopra per:

1. Impostare un numero massimo di tentativi di inserimento frase non collegato al numero massimo di lettere differenti presenti nella frase.

2. Riportare una frase con un *printf* che segnala al player quante lettere può inserire. Calcolare il numero di lettere da inserire come valore minimo tra numero massimo di inserimenti impostato al punto sopra e numero massimo di lettere differenti presenti nella frase.

3. Contare quanti tentativi di inserimento della frase sono già stati utilizzati e impostare un controllo sul numero massimo di tentati concessi in caso di raggiungimento del numero massimo numero di lettere inserite.

4. Tenere traccia delle lettere già inserite ed evitare che l'utente "sprechi" un nuovo inserimento lettera digitando una lettera già inserita.