



LUCA MANNA

EPICODE 2023/24

S2/L5 PROJECT
COMPITO S2/L5
1 DICEMBRE 2023

luca manna

S2/L5 PROJECT

Traccia:

Per agire come un Hacker bisogna capire come pensare fuori dagli schemi. L'esercizio di oggi ha lo scopo di allenare l'osservazione critica. Dato il codice in allegato, si richiede allo studente di:

1. Capire cosa fa il programma senza eseguirlo
2. Individuare dal codice sorgente le casistiche non standard che il programma non gestisce (esempio, comportamenti potenziali che non sono stati contemplati)
3. Individuare eventuali errori di sintassi / logici
4. Proporre una soluzione per ognuno di essi

Codice da valutare criticamente:

```
#include <stdio.h>

void menu ();
void moltiplica ();
void dividi ();
void ins_string();

int main ()
{
    char scelta = {'\0'};
    menu ();
    scanf ("%d", &scelta);

    switch (scelta)
    {
        case 'A':
```

Commented [Im1]: Come primo step analizziamo il codice in linguaggio C ed esaminiamo le componenti per poter al meglio affrontare la traccia e comprendere quali sono le discrepanze all'interno dei comandi presentati nell'esercizio.

Commented [Im2]: Come spiegato nella teoria il linguaggio C ha all'interno del suo sistema **librerie** e infatti questa istruzione fa parte della **libreria standard output/input in C**. Il file **stdio.h** fornisce le funzioni input/output standard come printf, scanf...

Commented [Im3]: La prima parte del codice in linguaggio C include le dichiarazioni delle funzioni (**menu**, **moltiplica**, **dividi** e **ins_string**). Ciò sta ad indicare al compilatore che le funzioni saranno definite in seguito nel codice.

Commented [Im4]: La funzione main.

1. **Char** scelta è una variabile del linguaggio C che viene inizializzata con il carattere nulla cioè **\0**.
2. **Scanf("%d", &scelta);** legge un solo carattere da input e viene memorizzato nella variabile scelta.
3. **Menu()** in questa funzione viene stampato presumibilmente un messaggio di benvenuto e con le opzioni del menu, come è stato eseguito ieri nell'esercizio per la creazione di un qui-game ideato con il linguaggio C.
4. **Switch-case:** come nella creazione del quiz-game nel precedente esercizio questa funzione consente al programma di eseguire una delle opzioni, se scelta sarà A la funzione si moltiplica(), se la scelta è B la funzione viene si divide e in caso di C la funzione viene detta ins_string.
5. **Return 0** come studiato in questi giorni il programma è eseguito correttamente e restituisce sempre 0.

```

        moltiplica();
        break;
        case 'B':
        dividi();
        break;
        case 'C':
        ins_string();
        break;
    }

    return 0;
}

void menu ()
{
    printf ("Benvenuto, sono un assistente digitale, posso aiutarti a
sbrigare alcuni compiti\n");
    printf ("Come posso aiutarti?\n");
    printf ("A >> Moltiplicare due numeri\nB >> Dividere due numeri\nC
>> Inserire una stringa\n");
}

void moltiplica ()
{
    short int a,b = 0;
    printf ("Inserisci i due numeri da moltiplicare:");
    scanf ("%f", &a);
    scanf ("%d", &b);

    short int prodotto = a * b;

    printf ("Il prodotto tra %d e %d e': %d", a,b,prodotto);
}

```

Commented [Im5]: In questa sezione del codice si definisce la funzione del menu che appunto ha la funzione di mostrare un messaggio di benvenuto e presentare le varie opzioni all'utente.

1. **Printf** Messaggio di benvenuto

2. **Printf** stampa la domanda "come posso aiutarti"

3. **Printf**("A>>Moltipli....") esegue le opzioni del menu con varie opzioni, **opzione A**, **opzione B**, **opzione C**. Questa funzione menu può essere utilizzata come interfaccia utente per selezionare diverse azioni.

Commented [Im6]: La funzione **moltiplica** consente all'utente di moltiplicare due numeri in modo tale da visualizzare il risultato. Vi sono anche qui delle discrepanze ma che analizzeremo durante l'esecuzione della traccia di oggi.

```
void dividi ()
{
    int a,b = 0;
    printf ("Inserisci il numeratore:");
    scanf ("%d", &a);
    printf ("Inserisci il denominatore:");
    scanf ("%d", &b);

    int divisione = a % b;

    printf ("La divisione tra %d e %d e': %d", a,b,divisione);
}
```

Commented [lm7]: Al contrario della sezione precedente, qui viene definita la **funzione dividi** che ha come scopo la divisione tra due numeri inseriti dall'utente: **printf**(inserisci il numeratore) **scanf**("%d", &a) legge un numero intero e lo assegna alla variabile a, **Printf** chiede all'utente di inserire il denominatore (qui è scritto denominatore), **scanf**("%d", &b) legge un numero intero e lo assegna alla variabile b, **int divisione = a % b** calcola divisione tra a e b assegnandolo a una variabile divisione di tipo int. **printf**("La divisione...") stampa il risultato tra i due numeri inseriti dall'utente e cioè il resto.

Dunque la funzione dividi consente all'utente di inserire due numeri calcolando il resto della divisione tra il numeratore e il denominatore in questo caso anche vi è una discrepanza che verrà segnalata in seguito.

```
void ins_string ()
{
    char stringa[10];
    printf ("Inserisci la stringa:");
    scanf ("%s", &stringa);
}
```

Commented [lm8]: La **funzione ins_string** ha come scopo di consentire all'utente l'inserimento di una stringa. **Char stringa[10]** capacità max 10 caratteri.

Printf("Inserisci...") consente all'utente appunto di inserire una stringa, **scanf**("%s"...") legge la stringa di caratteri da input e la memorizza nella array stringa ma anche qui vi è un errore.

Quindi la funzione **ins_string** consente all'utente di inserire una stringa di max 10 caratteri.

Potrebbe però verificarsi un problema di **OVERFLOW** nel caso in cui vengano digitati più di 10 caratteri(vedi soluzione).

Svolgimento:

1.

L'analisi del codice per comprendere cosa definisce senza eseguirlo è stata definita nei commenti del codice (a fianco) ma per fare chiarezza svolgerò punto per punto.

Un piccolo breve riassunto degli errori trovati e su cosa questo codice definisce senza eseguirlo:

- a. Dichiarazioni delle funzioni:
 - "Menu", "Moltiplica", "Dividi" e "ins_string"
- b. Funzione main:
 - "char scelta = {'\0'}",
 - "menu()", per visualizzare opzioni dell'utente,
 - "scanf("%d", &scelta)" questa funzione legge un numero intero da input e lo assegna a "scelta". In questo caso "scelta" essendo di tipo char ha un errore.

- c. Switch-Case: esecuzione delle opzioni scelta A (moltiplica), scelta B (dividi), scelta C (ins_string), return 0, termina restituendo 0.
 - d. Funzione menu: stampa un messaggio di benvenuto dando le varie opzioni del menu.
 - e. Funzione moltiplica: variabili "a", "b" e "short int", dove richiede all'utente di inserire due numeri e li trascrive nella variabile a e b. Inoltre calcola il prodotto tra la variabile a e la variabile b e lo stampa.
 - f. Funzione dividi: definisce le variabili "a", "b" di tipo "int". Chiede all'utente di inserire due numeri interi memorizzandoli in a e b e calcola il resto della divisione tra a e b e lo stampa.
 - g. Funzione **"ins_string"**: definisce una array di caratteri di stringa di dimensione 10. Dunque, richiede all'utente di inserire una stringa e la memorizza.
2. Il codice fornito non contempla e non individua le casistiche non standard o comportamenti potenziali nei seguenti punti:
- a. Nella funzione main il programma legge la scelta dell'utente con **"%d"** usando scanf ma "scelta" è dichiarato come "char", dunque andrebbe corretto con **"%c"** invece di **"%d"**.

int main ()

```
{
    char scelta = {'\0'};
    menu ();
    scanf ("%d", &scelta);    scanf ("%c", &scelta);

    switch (scelta)
    {
        case 'A':
            moltiplica();
            break;
```

- b. Nella funzione **"moltiplica"** la variabile "a" è dichiarata come **"short int"** e viene letta con **"%f"** che è adatto invece per i numeri in virgola. Dunque dovrebbe essere sostituito con **"%hd"** per leggere un **"short int"**.

void moltiplica ()

```
{
    short int a,b = 0;
```

```
printf ("Inserisci i due numeri da moltiplicare:");
scanf ("%f", &a); scanf ("%hd", &a);
scanf ("%d", &b);

short int prodotto = a * b;

printf ("Il prodotto tra %d e %d e': %d", a,b,prodotto);
}
```

- c. Nella funzione “dividi” la variabile “b” viene utilizzata come denominatore per la divisione ma il programma non può gestire nel caso in cui l’utente metterà 0 come tale e quindi se l’utente inserirà 0 come denominatore la divisione per 0 produrrà errori e non renderà dunque possibile l’operazione. In questo caso si deve aggiungere una condizione. Nell’ipotesi in cui un utente inserisca 0 al denominatore, si può aggiungere un loop “do-while” così che se l’utente inserisce il valore 0 al denominatore produrrà un messaggio di errore assicurandosi che l’utente inserirà sempre un numero diverso da 0.

Poi abbiamo un errore di sintassi “denumeratore” sostituito con “denominatore”

Inserisco anche la condizione “if”

e Mancanza di do {

void dividi ()

```
{
    int a, b = 0;
    printf ("Inserisci il numeratore:");
    scanf ("%d", &a);

    do{
        printf ("Inserisci il denumeratore:"); “denominatore diverso da zero:
    ”
        scanf ("%d", &b);

        if (b == 0) {
            printf("Errore: Il denominatore non può essere zero. Riprova.\n");
        }
    } while (b == 0);

    int divisione = a / b;
```

```
printf("La divisione tra %d e %d e': %d", a, b, divisione);
}
```

- d. Funzione **"ins_string"** l'array **"stringa"** è dimensioni 10 caratteri come è già stato definito in precedenza e dunque limita la possibilità all'utente di inserire una stringa più lunga di 10 caratteri causando un overflow del buffer.

Per dare la possibilità all'utente di inserire una stringa più lunga di 10 caratteri e prevenire così un overflow possiamo implementare la funzione **"scanf"** e pulire il buffer di input per evitare problemi con input successivi:

```
void ins_string ( )
{
char stringa[10]; char stringa[11]; // Aggiunto un elemento per il terminatore
printf("Inserisci la stringa (massimo 10 caratteri): ");
scanf("%10s", stringa);
printf ("Inserisci la stringa:"); // Pulizia del buffer di input
int c;
while ((c = getchar()) != '\n' && c != EOF);

// Ora la tua stringa contiene al massimo 10 caratteri
printf("Hai inserito la stringa: %s\n", stringa);
scanf ("%s", &stringa);
}
```

- Cambio di array da **"stringa"** a **"char stringa[11];"**
- Aggiunta di `scanf %10` per limitare la lettura a max 10 caratteri
- Aggiunta di `while` per pulire il buffer
- Stampa della stringa

- e. Il messaggio di benvenuto della funzione **"menu"** contiene un errore di battitura di **"assistente digitale"** poiché è scritto come **"assidente digitale"**

```
printf ("Benvenuto, sono un assidente digitale, posso aiutarti a sbrigare
alcuni compiti\n");
```

3. Errori di sintassi/logici

Errori di sintassi:

- a. `Char scelta = {'\0'};` `char scelta = '\0';`
- b. `Scanf ("%d", &scelta);` `scanf ("%c", &scelta)`
- c. `Char stringa [10]` quando si legge con `scanf` non è necessario utilizzare l'operatore `&`. Quindi sarà necessario cambiarlo con `scanf ("%s, stringa);`
- d. `Scanf ("%f", &a);` nella funzione `moltiplica` è stato dichiarato come `short int` e quindi si dovrà modificare con `scanf ("%hd", &a)` per leggere un `short int`.

Errori logici:

- a. Le variabili `a` e `b` sono state dichiarate nella funzione `moltiplica` come `short int` e `int` e potrebbe causare imprevisti durante l'esecuzione.
- b. Nella funzione `dividi` il programma calcola il resto della divisione (`a % b`). Se l'utente inserisce 0 come denominatore si verifica una divisione per 0 e potrà o non funzionare o portare a risultati imprevisti.
- c. Overflow del buffer di `stringa` nella funzione `ins_string` l'array `stringa` è limitato a 10 caratteri ma la lettura della `stringa` non è stata limitata. Quindi se l'utente inserirà più di 10 caratteri si verificherà un overflow del buffer.
- d. Uso errato dello `switch` infatti la variabile `scelta` è di tipo `char` e nei casi del blocco `switch` vengono utilizzati caratteri tra apici singoli tipo (`" 'A', 'B', 'C'`). Dunque, è più appropriato usare `switch ((int)scelta)` o più semplicemente `switch (scelta)` con casi come `'A'`.

4. Soluzione:

COMANDI C PROPOSTI	COMANDI C RISOLTI
<pre>#include <stdio.h> void menu (); void moltiplica (); void dividi (); void ins_string(); int main () { char scelta = {'\0'}; menu (); scanf ("%d", &scelta); switch (scelta) { case 'A': moltiplica (); break; case 'B': dividi (); break; case 'C': ins_string (); break; } return 0; }</pre>	<pre>#include <stdio.h> void menu (); void moltiplica (); void dividi (); void ins_string (); int main() { char scelta = '\0'; menu (); scanf ("%c", &scelta); switch (scelta) { case 'A': moltiplica (); break; case 'B': dividi (); break; case 'C': ins_string (); break; } return 0; } void menu () {</pre>

Commented [Im9]: Correzione inizializzazione.

Commented [Im10]: Correzione del formato per leggere un carattere.

```
void menu ()
{
    printf ("Benvenuto, sono un
    assistente digitale, posso aiutarti a
    sbrigare alcuni compiti\n");
    printf ("Come posso
    aiutarti?\n");
    printf ("A >> Moltiplicare
    due numeri\nB >> Dividere due
    numeri\nC >> Inserire una
    stringa\n");
}
```

```
void moltiplica ()
{
    short int a, b = 0;
    printf ("Inserisci i due
    numeri da moltiplicare:");
    scanf ("%f", &a);
    scanf ("%d", &b);

    short int prodotto = a * b;

    printf ("Il prodotto tra %d e
    %d e': %d", a,b,prodotto);
}
```

```
void dividi ()
{
    int a,b = 0;
    printf ("Inserisci il
    numeratore:");
    scanf ("%d", &a);
    printf ("Inserisci il
    denominatore:");
    scanf ("%d", &b);
```

```
printf("Benvenuto, sono un assistente digitale,
posso aiutarti a sbrigare alcuni compiti\n");
printf("Come posso aiutarti?\n");
printf("A >> Moltiplicare due numeri\nB >>
Dividere due numeri\nC >> Inserire una
stringa\n");
}
```

```
void moltiplica()
{
    short int a, b = 0;
    printf("Inserisci i due numeri da
    moltiplicare:");
    scanf("%hd", &a);
    scanf("%hd", &b);
```

```
short int prodotto = a * b;

printf("Il prodotto tra %hd e %hd e': %hd",
a, b, prodotto);
}
```

```
void dividi ()
{
    int a, b = 0;
    printf("Inserisci il numeratore:");
    scanf("%d", &a);

    do {
        printf("Inserisci il denominatore diverso
        da zero:");
        scanf("%d", &b);

        if (b == 0) {
```

Commented [lm11]: Correzione errore di sintassi
assistente—>assistente

Commented [lm12]: Correzione per leggere un short int.

Commented [lm13]: Correzione per stampare un short int

<pre> int divisione = a % b; printf ("La divisione tra %d e %d e': %d", a,b,divisione); void ins_string () { char stringa[10]; printf ("Inserisci la stringa:"); scanf ("%s", &stringa); </pre>	<pre> printf("Errore: Il denominatore non può essere zero. Riprova.\n"); } } while (b == 0); int divisione = a / b; printf("La divisione tra %d e %d e': %d", a, b, divisione); } void ins_string () { char stringa[11]; printf("Inserisci la stringa (massimo 10 caratteri): "); scanf("%10s", stringa); int c; while ((c = getchar()) != '\n' && c != EOF); printf("Hai inserito la stringa: %s\n", stringa); } </pre>
---	---

Commented [lm14]: Denominatore diverso da 0 per dare errore in caso l'utente inserisca 0.

Commented [lm15]: Pulizia del buffer input per non creare OVERFLOW.

Commented [lm16]: Correzione per risolvere la stringa max 10 caratteri per limitare gli errori di riproduzione.

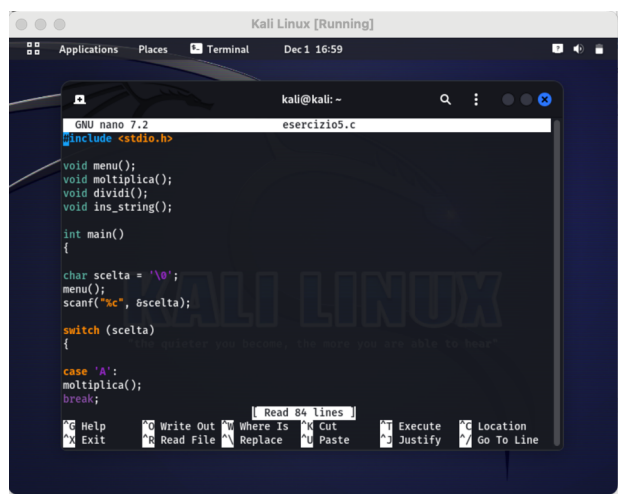
Trascrizione nel nuovo comando su Kali Linux:

Per comprovare l'effettività delle correzioni apportate ai comandi in C dell'esercizio proposto ho deciso di trascriverli su Kali Linux.

1. Ho eseguito i comandi per trascrivere il file:

"nano esercizio5.c"

2. Ho modificato l'editor di testo (vedi Tab.1)



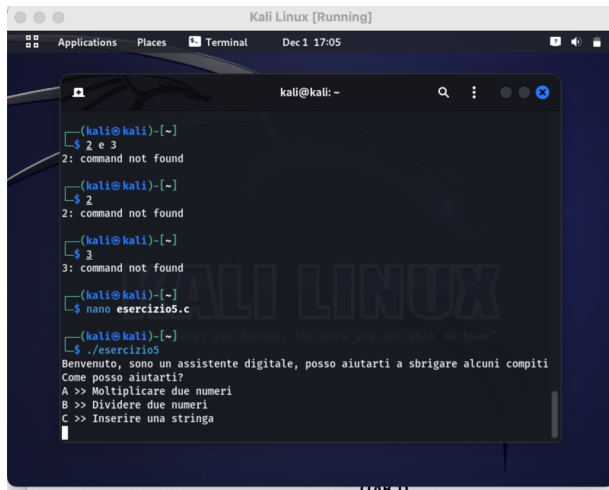
```
GNU nano 2.2.8 esercizio5.c
#include <stdio.h>

void menu();
void moltiplica();
void dividi();
void ins_string();

int main()
{
    char scelta = '\0';
    menu();
    scanf("%c", &scelta);
    switch (scelta)
    {
        case 'A':
            moltiplica();
            break;
    }
}
```

(TAB.1)

3. Dunque, dopo la trascrizione ho verificato che l'esercizio5.c fosse sulla giusta directory con il comando **"ls"** e ho eseguito i comandi di compilazione ed esecuzione **"gcc -o esercizio5.c"** e **"./esercizio5"**. (Vedi Tab.2)

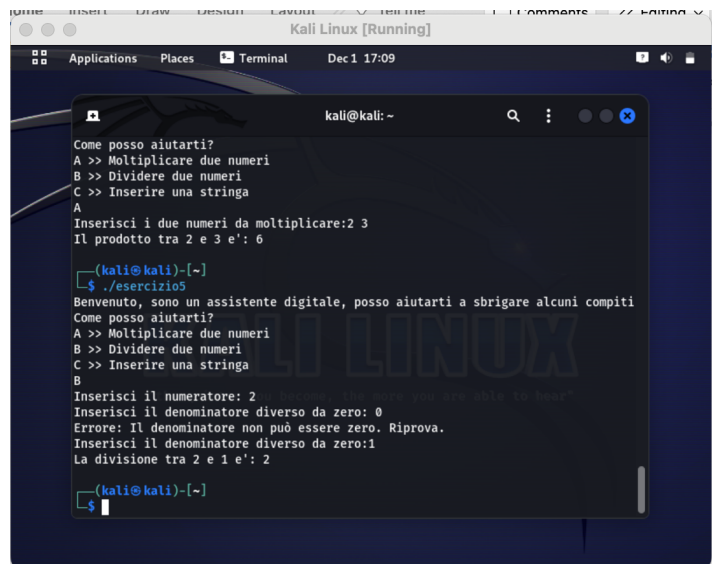


```
kali@kali:~$ 2 e 3
2: command not found
kali@kali:~$ 2
2: command not found
kali@kali:~$ 3
3: command not found
kali@kali:~$ nano esercizio5.c
kali@kali:~$ ./esercizio5
Benvenuto, sono un assistente digitale, posso aiutarti a sbrigare alcuni compiti
Come posso aiutarti?
A >> Moltiplicare due numeri
B >> Dividere due numeri
C >> Inserire una stringa
```

(TAB.2)

L' esercizio5 funziona con le modifiche apportate ora verifico se vi sono ulteriori problemi con l'esecuzione del programma.

4. Come si evince dalle TAB.3-4 il programma sembra funzionare correttamente con le modifiche al denominatore e alle stringhe. Infatti, quando ho provato ad inserire il denominatore 0 mi ha dato un errore e mi ha dato la possibilità di ridigitare un numero diverso da 0. Inoltre, ha corretto anche la stringa quando ho provato ad inserire più di 10 caratteri ne ha riportati solo 10.



```

Kali Linux [Running]
Applications Places Terminal Dec 1 17:09

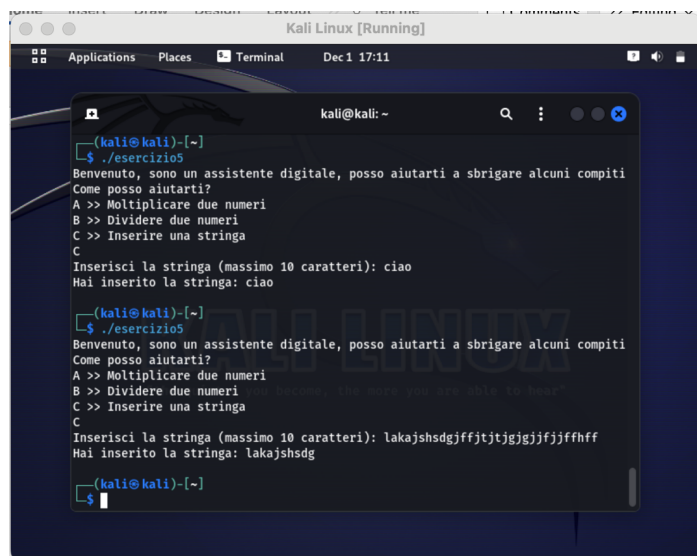
kali@kali: ~
Come posso aiutarti?
A >> Moltiplicare due numeri
B >> Dividere due numeri
C >> Inserire una stringa
A
Inserisci i due numeri da moltiplicare: 2 3
Il prodotto tra 2 e 3 e': 6

(kali@kali)-[~]
$ ./esercizio5
Benvenuto, sono un assistente digitale, posso aiutarti a sbrigare alcuni compiti
Come posso aiutarti?
A >> Moltiplicare due numeri
B >> Dividere due numeri
C >> Inserire una stringa
B
Inserisci il numeratore: 2
Inserisci il denominatore diverso da zero: 0
Errore: Il denominatore non può essere zero. Riprova.
Inserisci il denominatore diverso da zero: 1
La divisione tra 2 e 1 e': 2

(kali@kali)-[~]
$

```

(TAB.3)



```

Kali Linux [Running]
Applications Places Terminal Dec 1 17:11

kali@kali: ~
(kali@kali)-[~]
$ ./esercizio5
Benvenuto, sono un assistente digitale, posso aiutarti a sbrigare alcuni compiti
Come posso aiutarti?
A >> Moltiplicare due numeri
B >> Dividere due numeri
C >> Inserire una stringa
C
Inserisci la stringa (massimo 10 caratteri): ciao
Hai inserito la stringa: ciao

(kali@kali)-[~]
$ ./esercizio5
Benvenuto, sono un assistente digitale, posso aiutarti a sbrigare alcuni compiti
Come posso aiutarti?
A >> Moltiplicare due numeri
B >> Dividere due numeri
C >> Inserire una stringa
C
Inserisci la stringa (massimo 10 caratteri): lakajshsdgjffjtjtjtgjjfjffhff
Hai inserito la stringa: lakajshsdg

(kali@kali)-[~]
$

```

(TAB.4)

Conclusioni e Raccomandazioni sul Progetto S2/L5

Dopo lo svolgimento di questo progetto si deduce che:

è fondamentale comprendere appieno il codice che si sta scrivendo, assicurandosi di capire ogni parte del programma, dalle dichiarazioni delle funzioni alle singole istruzioni.

Un punto cruciale è la gestione degli input utente, bisogna considerare tutte le possibili casistiche, validare ogni input e controllare i messaggi di errore durante l'esecuzione poiché sono significativi in caso di inserimenti non validi.

Inoltre, si devono verificare attentamente tutti passaggi di compilazione ed esecuzione, assicurandosi di essere nella directory corretta e che l'ambiente di sviluppo sia configurato correttamente.

Sfruttare le risorse online, come la documentazione del linguaggio C e forum di programmazione. Queste risorse possono essere utili per risolvere problemi specifici e approfondire la comprensione del linguaggio in particolare per utenti come me che si avvicinano per la prima volta a questo tipo di linguaggio.

Un altro aspetto fondamentale è la strutturazione modulare dei codici, utilizzando funzioni per gestire compiti specifici così da rendere il codice più leggibile facilitando così la risoluzione dei problemi.

Prima di considerare il codice completo, è necessario eseguire test per assicurarsi che tutte le funzionalità siano implementate correttamente e che il programma gestisca ogni possibile input in modo robusto.

Infine, i codici sono sempre tanti e tutti da scoprire quindi è necessario chiedere aiuto se i programmi producono bug o non eseguono correttamente il programma.

ESERCIZIO S2/L5

Traccia:

Riprendete il codice del programma che avete scritto e pensiamo all'ottimizzazione del codice alla gestione delle situazioni non previste e facciamo le seguenti considerazioni:

1. Cosa succede se l'utente inserisce una lettera diversa da A o B in fase di scelta iniziale? Il programma termina, ma non è una casistica che abbiamo gestito.
2. Cosa succede se l'utente inserisce un nome che ha più caratteri della dimensione dell'array «nome» che abbiamo dichiarato inizialmente nella fase di avvio nuova partita? Riceveremo un errore (provate ad inserire una sequenza molto lunga di caratteri)
3. Cosa succede se l'utente inserisce la lettera D per la risposta alle domande durante una partita? O un carattere numerico? Tutte queste situazioni vanno considerate in fase di programmazione in quanto errori logici o errori di mancata gestione di situazioni non standard potrebbero portare a bug nel codice che potrebbero essere sfruttati da un attaccante per prendere controllo dell'esecuzione del programma ed eseguire codice malevolo.

Traccia:

Riprendete il programma scritto in precedenza e identificate tutte le casistiche non contemplate.

Provate a proporre un modello per gestirle modificando il codice sorgente del vostro programma.

Aiutatevi pure con le risorse online, piccolo aiuto: cercate come gestire in maniera sicura l'input dell'utente (soprattutto quando parliamo di stringhe).

Per gestire il caso in cui l'utente inserisce una lettera diversa da A o B si può introdurre una logica per dare un messaggio di errore.

Inoltre, è necessario implementare controlli più robusti sull'input dell'utente.

In particolare, è necessario utilizzare funzioni e tecniche che prevengano errori di accesso fuori dai limiti degli array e che gestiscano gli imprevisti in modo sicuro.

Codici gioco precedenti	Codici gioco aggiornato
<pre>#include <stdio.h> int main() { int scelta; char nome[50]; int punteggio = 0; printf("Benvenuto al gioco di domande!\n"); printf("Scopo del gioco: rispondere correttamente alle domande e ottenere il massimo punteggio.\n"); do { printf("\nMenu di scelta:\n"); printf("A) Iniziare una nuova partita\n"); printf("B) Uscire dal gioco\n"); printf("Scelta: "); scanf(" %c", &scelta); if (scelta == 'A' scelta == 'a') { printf("\nInserisci il tuo nome: "); scanf("%s", nome); // Set di domande printf("\nDomanda 1: Qual è la capitale dell'Italia?\n"); printf("A) Roma (+2 punti)\nB) Parigi\nC) Berlino\nScelta: "); char risposta1;</pre>	<pre>#include <stdio.h> #include <stdlib.h> #include <string.h> #define int main() { int scelta; char nome[MAX_NOME_LENGTH]; int punteggio = 0; printf("Benvenuto al gioco di domande!\n"); printf("Scopo del gioco: rispondere correttamente alle domande e ottenere il massimo punteggio.\n"); do { printf("\nMenu di scelta:\n"); printf("A) Iniziare una nuova partita\n"); printf("B) Uscire dal gioco\n"); printf("Scelta: "); if (scanf(" %c", &scelta) != 1 (scelta != 'A' && scelta != 'a' && scelta != 'B' && scelta != 'b')) { printf("Scelta non valida. Inserire A o B. Riprova.\n"); caso di input non valido. while (getchar() != '\n'); continue; //</pre>

Commented [lm17]: La modifica è stata apportata per gestire l'input dell'utente nella fase iniziale garantendo la convalida dell'opzione a,A,B,b. Se l'utente inserisse una lettera diversa, il programma emette un messaggio di errore!

<pre> scanf("%c", &risposta1); if (risposta1 == 'A' risposta1 == 'a') { punteggio += 2; printf("Risposta corretta! + %d punti\n", 2); } else { printf("Risposta errata. Nessun punto aggiunto.\n"); } printf("\nDomanda 2: Quale pianeta è conosciuto come la 'stella del mattino'?\n"); printf("A) Marte\nB) Venere (+3 punti)\nC) Giove\nScelta: "); char risposta2; scanf("%c", &risposta2); if (risposta2 == 'B' risposta2 == 'b') { punteggio += 3; printf("Risposta corretta! + %d punti\n", 3); } else { printf("Risposta errata. Nessun punto aggiunto.\n"); } // Nuove domande // ... // Risultato finale printf("\nPartita completata, %s!\n", nome); printf("Il tuo punteggio è: %d\n", punteggio); } else if (scelta != 'B' && scelta != 'b') { printf("Scelta non valida. Riprova.\n"); } </pre>	<pre> } if (scelta == 'A' scelta == 'a') { printf("\nInserisci il tuo nome: "); if (scanf("%49s", nome) != 1) { printf("Errore di input. Terminazione del programma.\n"); exit(EXIT_FAILURE); } } } while (scelta != 'B' && scelta != 'b'); printf("\nGrazie per aver giocato!\n"); return 0; } </pre>
---	--

Commented [lm18]: Per prevenire un buffer overflow nel caso in cui venga inserito un nome troppo lungo, questa funzione limita la lunghezza massima del nome con "%49s" nel scanf.

<pre> } while (scelta != 'B' && scelta != 'b'); printf("\nGrazie per aver giocato!\n"); return 0; } </pre>	
--	--

1. CASO IN CUI VIENE INSERITA LA LETTERA D, dopo le nuove implementazioni se l'utente digita una lettera diversa da A o B, il programma produrrà un messaggio di errore. (TAB.1)
2. Ho provato ad inserire un nome con un carattere più lungo di 50 caratteri però il gioco continua a farmi procedere. Ho re-inserito più una nuova funzione che potesse ovviare al problema e alla fine NON sono riuscito a trovare una funzione logica per immettere un limite al carattere max di 50 per la scelta del nome (TAB.2)

```

Kali Linux [Running]
Applications Places Terminal Dec 1 18:42
kali@kali: ~
Domanda 5: Quanti studenti riescono a sopravvivere dopo un corso in cybersecuri
ty epicode?
A) Nessuno, credetemi!
B) Rimarranno solo alcuni superstiti
C) Ne rimarranno solo 2
Scelta: a
Risposta corretta! +800 punti

Partita completata,lucaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
daaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaa!
Il tuo punteggio è:1633772770

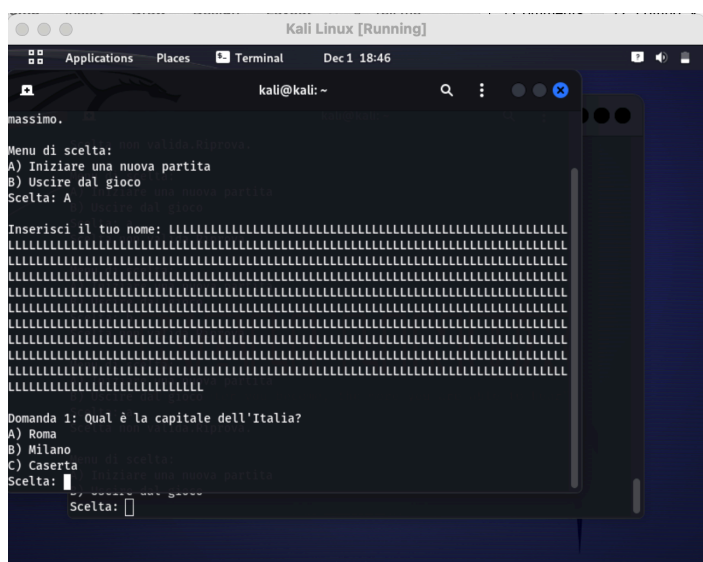
Menu di scelta:
A) Iniziare una nuova partita
B) Uscire dal gioco
Scelta: D
Scelta non valida.Riprova.

Menu di scelta:
A) Iniziare una nuova partita
B) Uscire dal gioco
Scelta:

```

(TAB.1)

LUCA MANNA



(TAB.2)