**BUG Hunting S2/L5**

Zhongshi Liu

19/4/2024

**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:

● Capire cosa fa il programma senza eseguirlo.

● Individuare dal codice sorgente le casistiche non standard che il programma non gestisce (esempio, comportamenti potenziali che non sono stati contemplati).

● Individuare eventuali errori di sintassi / logici.

● Proporre una soluzione per ognuno di essi.

Descrizione della Programma

Questo programma in C è un semplice assistente digitale che fornisce un menu con tre opzioni:

1)Moltiplicare due numeri.

2)Dividere due numeri.

3)Inserire una stringa.

All'avvio del programma, viene visualizzato il menu con le opzioni disponibili. L'utente può quindi selezionare una delle opzioni inserendo la lettera corrispondente. Dopo aver inserito la scelta, il programma esegue la funzione corrispondente.

● La funzione moltiplica() chiede all'utente di inserire due numeri interi, quindi calcola il loro prodotto e lo stampa a schermo.

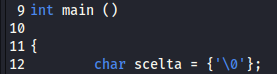
● La funzione dividi() chiede all'utente di inserire due numeri interi, quindi calcola il resto della divisione tra il primo e il secondo numero e lo stampa a schermo.

● La funzione ins\_string() chiede all'utente di inserire una stringa di massimo 10 caratteri e la memorizza in una variabile stringa.

Tuttavia, ci sono alcuni problemi nel codice.

**Errori**

**Errore N.1**



1) Nella dichiarazione char scelta = {'\0'}; , {} non è necessario. Basta char scelta = '\0'; .

2)All'inizio del programma, la variabile scelta viene inizializzata con il carattere nullo, ma viene sovrascritta dalla scansione dell'input utente. Potrebbe essere eliminata l'inizializzazione.

--------------------------------------------------------------------------------------------------------

**Errore N.2**

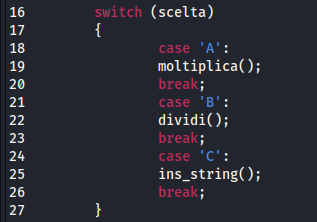


1) **Errore di sintassi**(riga.14): Il tipo di dato utilizzato per la variabile 'scelta' è 'char', tuttavia, nella scansione (scanf) viene specificato '%d', che è per interi. Dovrebbe essere utilizzato '%c' per leggere un carattere.

2) È preferibile inserire uno spazio prima del %c (" %c") per evitare potenziali problemi dovuti al buffering del sistema operativo.

--------------------------------------------------------------------------------------------------------

**Errore N.3**

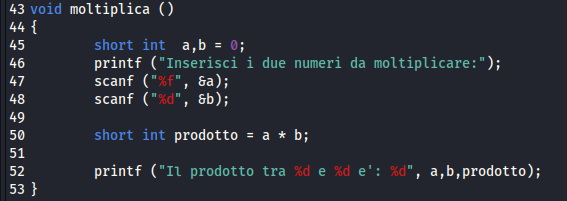


1) Nel linguaggio C, le lettere maiuscole e minuscole sono considerate diverse. Pertanto, se l'utente inserisce 'a', 'b' o 'c', il programma interrompe l'esecuzione anziché procedere. Sarebbe opportuno aggiungere altri case con i caratteri minuscoli.

2) Quando utende inserisce un valore fuori opzione il programma si termina, e' giusto includere una gestione default che comunichi all'utente che l'input non è valido e gli consenta di ripetere l'inserimento.

--------------------------------------------------------------------------------------------------------

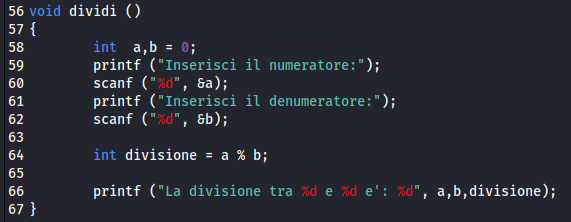
**Errore N.4**



**Errore di sintassi**(righe(45,47,48 e 52)): Le variabili a, b, e prodotto sono di tipo short int. Pertanto, gli identificatori corretti da utilizzare con scanf e printf devono essere %hd.

--------------------------------------------------------------------------------------------------------

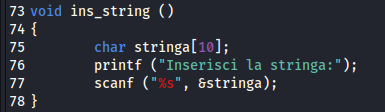
**Errore N.5**



1. **Casistica non gestita**: quando l'utente inserisce 0, il programma restituisce un errore poiché il denominatore non può essere 0.
2. **Errore logico** (riga 64): l'operatore % restituisce il resto della divisione; per eseguire l'operazione di divisione, è necessario utilizzare il simbolo /.
3. **Casistica non gestita**(riga.64): il risultato di un'operazione di divisione è spesso un numero decimale; sarebbe corretto utilizzare una variabile di tipo float.

--------------------------------------------------------------------------------------------------------

**Errore N.6**

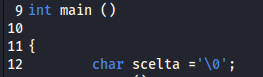


1. **Errore di sintassi**(riga.77): Le stringhe in C sono trattate come array di char e quindi la variabile è un puntatore a sé stante. Di conseguenza, non è necessario inserire '&' prima del nome della variabile, in questo caso la variabile 'stringa'.
2. **Casistica non gestita**(riga.77): Quando l'utente inserisce più di 10 caratteri, il programma va in stack overflow. Per risolvere questo problema, è necessario leggere solo i primi 9 caratteri digitati dall'utente specificando %9s.

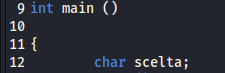
---------------------------------------------------------------------------------------------------------

**Risoluzione**

**Risoluzione N.1**



1) Nella dichiarazione char scelta = {'\0'}; , {} non è necessario. Basta char scelta = '\0'; .



2) ll'inizio del programma, la variabile scelta viene inizializzata con il carattere nullo, ma viene sovrascritta dalla scansione dell'input utente. Potrebbe essere eliminata l'inizializzazione.

--------------------------------------------------------------------------------------------------------

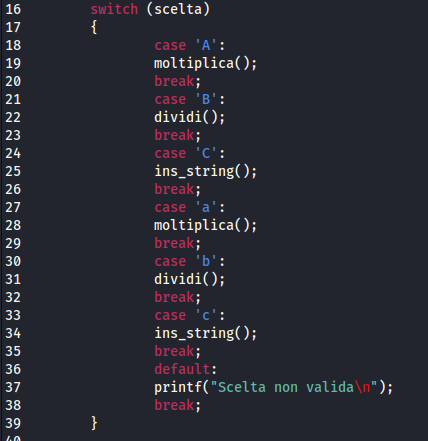
**Risoluzione N.2**



**Risoluzione**(riga.14): Il tipo di dato utilizzato per la variabile 'scelta' è 'char', tuttavia, di consequenza si utilizza “ %c” per leggere un carattere.

--------------------------------------------------------------------------------------------------------

**Risoluzione N.3**



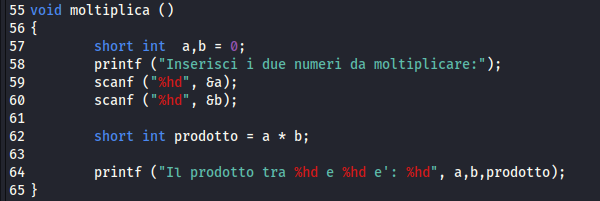
1) **Risoluzione**: Nel linguaggio C, le lettere maiuscole e minuscole sono considerate diverse. Pertanto, se l'utente inserisce 'a', 'b' o 'c', il programma interrompe l'esecuzione anziché procedere. Sarebbe opportuno aggiungere altri case con i caratteri minuscoli.



2) **Risoluzione**: Quando utende inserisce un valore fuori opzione il programma si termina, e' giusto includere una gestione default che comunichi all'utente che l'input non è valido e gli consenta di ripetere l'inserimento.

-----------------------------------------------------------------------------------------------------------------------------------------

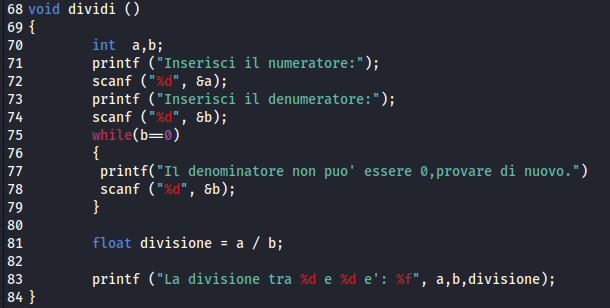
**Risoluzione N.4**

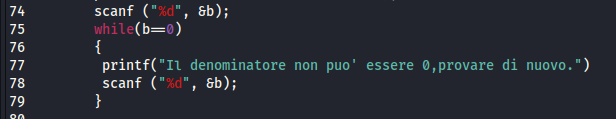


**Risoluzione**(righe(57,59,60 e 64)): Le variabili a, b, e prodotto sono di tipo short int. Pertanto, gli identificatori corretti da utilizzare con scanf e printf devono essere %hd.

--------------------------------------------------------------------------------------------------------

**Risoluzione N.5**





1. **Risoluzione Casistica non gestita**: È necessario aggiungere una condizione che impedisca la divisione per 0 e permetta all'utente di inserire un altro numero. In questo caso, ho utilizzato un ciclo while con la condizione (b == 0), il quale richiederà all'utente di reinserire un valore diverso da 0 nel caso in cui venga inserito 0 come valore per la variabile b.

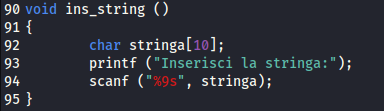


1. **Risoluzione dell’errore logico**: l'operatore % restituisce il resto della divisione; per eseguire l'operazione di divisione, è necessario utilizzare il simbolo /.

3) **Risoluzione** **Casistica non gestita**: il risultato di un'operazione di divisione è spesso un numero decimale; sarebbe corretto utilizzare una variabile di tipo float.

--------------------------------------------------------------------------------------------------------

**Risoluzione N.6**



1. **Risoluzione dell’errore di sintassi**(riga.94): Le stringhe in C sono trattate come array di char e quindi la variabile è un puntatore a sé stante. Di conseguenza, non è necessario inserire '&' prima del nome della variabile, in questo caso la variabile 'stringa'.
2. **Risoluzione Casistica non gestita**(riga.94): Quando l'utente inserisce più di 10 caratteri, il programma va in stack overflow. Per risolvere questo problema, è necessario leggere solo i primi 9 caratteri digitati dall'utente specificando %9s.

-------------------------------------------------------------------------------------------------------------------------------------------

**Il codice completo e corretto**

