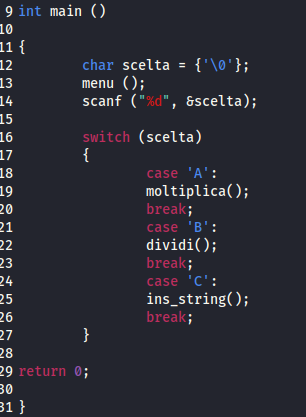
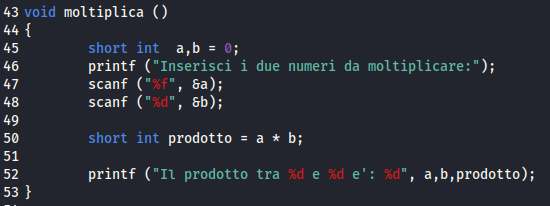
Il programma in questione consiste in un assistente digitale capace di eseguire alcune semplici operazioni, come la moltiplicazione/divisione tra due numeri e l’inserimento di una stringa.

Vediamo ora nel dettaglio gli errori logico/sintattici riscontrati nel programma:



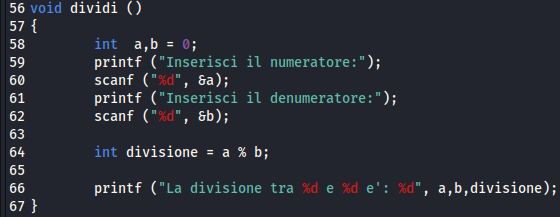
Errore n.1: nella funzione main viene utilizzato un costrutto di tipo ‘Switch’ ma al suo interno non viene definito il ‘default’, ovvero non sono state gestite tutte quelle casistiche in cui l’utente scelga una risposta diversa da ‘A, B o C’.



Errore n.2 e n.3:

Rigo 47: la variabile ‘&f’ nella funzione ‘scanf’ è errata in quanto non fa riferimento alla variabile con cui ‘a’ e ‘b’ sono state dichiarate, ovvero ‘short int’, quindi numeri interi. La variabile ‘%f’ è di tipo ‘float’.

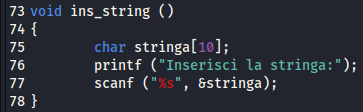
Altra problematica sta proprio nell’utilizzo della variabile ‘short int’ in quanto, essendo un numero di 2 byte (da -32.000 circa a +32.000 circa) potrebbe non rientrare nel range degli interi scelti per il prodotto dei due numeri.



Errore n.4, n.5:

Rigo 64: la variabile ‘int’ per la funzione ‘dividi’ è errata, visto che l’operazione potrebbe dare come risultato anche numeri decimali. Inoltre viene erroneamente rappresentata con ‘%’, simbolo invece utilizzato per un’altra operazione, il modulo.

Altro errore è quello di non aver considerato nella funzione la divisione per 0, in quanto genera un errore a livello di istruzioni (‘floating point exception’).



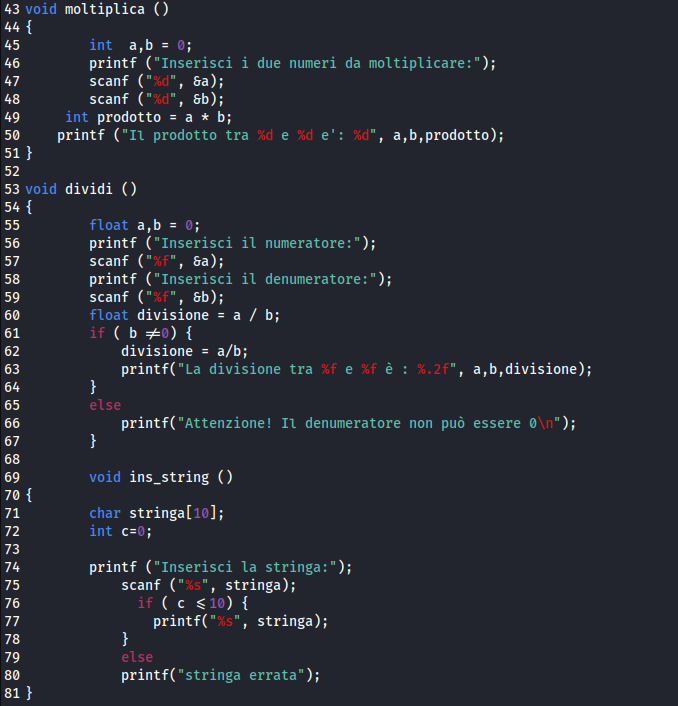
Errore n.6:

nella funzione ‘ins\_string’, essendo dichiarata una variabile ‘char’ di 10 caratteri, manca un controllo su di esso, ovvero sulla massima dimensione dell’input utente. Infatti, se l’utente decidesse di immettere più di 10 caratteri, si andrebbe a generare un ‘overflow’, una pericolosa vulnerabilità che, se sfruttata da un attaccante o criminale informatico, potrebbe immettere all’interno del programma un codice malevolo, prendendone il controllo.

Codice Corretto:

Immagine che contiene testo, schermata

Descrizione generata automaticamente



Rigo 28: ho aggiunto la variabile di ‘default’ al costrutto ‘switch’ così da gestire le casistiche in cui l’utente non scelga una delle tre opzioni consigliate, facendo stampare con la funzione ‘printf’ la scritta sopra indicata al rigo 29.

Rigo 45: ho tolto la variabile ‘short int’ preferendo al suo posto quella di ‘int’ per le motivazioni spiegate precedentemente.

Rigo 47: ho sostituito la variabile ‘%f’ con ‘%d’, essendo essa riferita a quella di tipo ‘int’.

Rigo 55: utilizzo la variabile ‘float’ (numeri reali) sia per ‘a’ che per ‘b’ così da avere un risultato preciso nella divisione, messa anch’essa in relazione alla variabile ‘float’(Rigo 60).

Rigo 57 e 59: utilizzo la variabile ‘%f’ nella funzione ‘scanf’ perché di tipo ‘float’.

Rigo 61-67: in questo blocco di codice utilizzo il costrutto ‘if-else’, un’istruzione di controllo condizionale, per gestire la divisione per 0. Se il denumeratore sarà diverso da 0, allora la divisione darà esito positivo (utilizzo la variabile’%.2f’ nel rigo 63 per pura comodità, prendendo in considerazione due cifre oltre il punto), altrimenti, tramite ‘printf’, comparirà la scritta sopra indicata al rigo 66.

Rigo 72: dichiaro una variabile ‘int c’ per indicare i caratteri della variabile ‘char stringa [10]’.

Rigo 76-81: Per controllare la dimensione dell’input utente ho pensato di utilizzare ancora un costrutto di tipo ‘if-else’: se il carattere della stringa è minore o uguale a 0, allora verrà stampata correttamente, in caso contrario non verrà accettata.