Progetto S2L5

Obiettivi:

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

Risoluzione:

Punto 1:

Il programma consiste in un menù iniziale in cui viene chiesto all'utente di effettuare una scelta tra moltiplicare due numeri, dividerli oppure inserire una stringa.
L'utente quindi deve digitare A, B o C per accedere alla funzione desiderata.

Punto 2:

Il codice sorgente non gestisce:

- il caso in cui un utente non scriva una delle tre lettere indicate per accedere alle funzioni;
- Il fatto che l'utente possa inserire la lettera corrispondente ma minuscola;
- il caso in cui l'utente inserisca un valore non valido per la funzione di moltiplicazione e di divisione;
- il caso in cui l'utente ottenga un risultato fuori range della variabile short nel caso della moltiplicazione;
- il caso in cui l'utente inserisca 0 come denominatore nella divisione;
- il caso in cui l'utente inserisca una stringa con più di dieci caratteri nella funzione di inserimento stringa;

Punto 3 e 4:

Si parta dal presupposto che il codice allegato è stato inserito in un compilatore e di conseguenza ogni riga numerata, dove la riga 1 è identificata da **#include <stdio.h>**. Gli errori verranno quindi segnalati facendo riferimento al numero di riga corrispondente e verrà proposta la soluzione:

- riga 14: il valore tra apici è “%d” ma deve essere “%c”;
- riga 27: manca il default nello switch, non è obbligatorio metterlo ma andrebbe aggiunto per gestire il caso in cui la lettera non sia una di quelle dei casi precedenti;

- riga 47: il valore tra apici è “%f” ma deve essere “%d” in quanto variabile short;
- riga 47,48: non viene effettuato nessun controllo sul fatto che i valori inseriti siano numeri. Per ovviare a questo problema importerei la libreria **<ctype.h>** e utilizzerei la funzione **isdigit()** per controllare l'input dell'utente. Metterei il check all'interno di un do while così l'utente rimarrà bloccato in quel punto fino a che non inserirà un valore corretto;
- riga 50: la variabile prodotto è dichiarata come short int ma sarebbe meglio dichiararla a int in caso di risultati oltre il range dello short;
- riga 60, 62: vedi riga 47,48;
- riga 62: all'inserimento del denumeratore è necessario gestire il caso in cui un utente inserisca lo 0. In quel caso utilizzerei un do while per continuare a chiedere un valore valido all'utente che sia diverso da 0.
- riga 64: l'operatore utilizzato è il modulo % ma va utilizzata la divisone /. La variabile è dichiarata come int ma va dichiarata come float poiché con una divisione è facile avere un numero che non sia intero;
- riga 66: l'indicatore del risultato della divisione è “%d” ma avendo detto che è meglio sostituirlo con un float dovremmo avere “%f”;
- riga 77: la stringa è indicata con &stringa ma va chiamata senza la & perché viene trattata come un puntatore dalla funzione scanf. Per verificare che una stringa sia tale è necessario utilizzare la funzione fgets che controlla la stringa per il numero di caratteri che gli diciamo. In questo modo andremo ad evitare lo stack overflow.

Davide Lecci