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:

```
14      scanf ("%d", &scelta);
```

il valore tra apici è “%d” ma deve essere “%c”;

```
14      scanf ("%c", &scelta);
```

riga 16:

```
16      switch (scelta)
17      {
18          case 'A':
19              moltiplica();
20              break;
21          case 'B':
22              dividi();
23                  break;
24          case 'C':
25                  ins_string();
26                  break;
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;

```
switch (scelta)
{
    case 'A':
        moltiplica();
        break;
    case 'B':
        dividi();
        break;
    case 'C':
        ins_string();
        break;
    default:
        printf("La scelta inserita non è valida!");
        break;
}
```

riga 47:

```
47      scanf ("%f", &a);
```

il valore tra apici è “%f” ma deve essere “%d” in quanto variabile short;

```
47      scanf ("%d", &a);
```

riga 47,48:

```
47      scanf ("%d", &a);
48      scanf ("%d", &b);
```

Per ovviare a questo problema è necessario assegnare il valore ritornato dalla funzione scanf, questa funzione ritorna 1 se il valore inserito è corretto. Quindi si inserisce un loop do while che andrà avanti finché l'utente non inserirà i numeri corretti.

Naturalmente bisogna anche pulire il buffer dell'input altrimenti il ciclo continuerà a stampare all'infinito e non aspetterà l'input dell'utente.

```
47      int resultA, resultB;
48      do {
49          printf("Inserisci due numeri da moltiplicare: ");
50          resultA = scanf("%d", &a);
51          resultB = scanf("%d", &b);
52
53          if (resultA != 1 || resultB != 1) {
54              printf("Input non valido. Per favore inserisci due numeri interi.\n");
55              while (getchar() != '\n');
56          }
57      } while (resultA != 1 || resultB != 1);
```

riga 50:

```
50      short int prodotto = a * b;
```

la variabile prodotto è dichiarata come short int ma sarebbe meglio dichiararla a int in caso di risultati oltre il range dello short.

```
50      int prodotto = a * b;
```

riga 60,62:

```
60          scanf ("%d", &a);
61          printf ("Inserisci il denumeratore:");
62          scanf ("%d", &b);
```

stessa soluzione applicata per riga 47,48.

riga 62:

```
62          scanf ("%d", &b);
```

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.

```
59      int resultA, resultB;
60      do {
61          printf("Inserisci due numeri da dividere: ");
62          resultA = scanf("%d", &a);
63          resultB = scanf("%d", &b);
64
65          if (resultA != 1 || resultB != 1) {
66              printf("Input non valido. Per favore inserisci due numeri interi.\n");
67              while (getchar() != '\n');
68          } else if (b == 0) {
69              printf("Input non valido. Il secondo numero non può essere zero.\n");
70          }
71      } while (resultA != 1 || resultB != 1 || b == 0);
```

riga 64:

```
64          int divisione = a % b;
```

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;

```
64          float divisione = a % b;
```

riga 66:

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

l'indicatore del risultato della divisione è "%d" ma avendo detto che è meglio sostituirlo con un float dovremmo avere "%f";

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

riga 77:

```
77          scanf ("%s", &stringa);
```

la stringa è indicata con &stringa ma va chiamata senza la & perché viene trattata come un puntatore dalla funzione scanf.

Per verificare che la stringa non ecceda il numero di caratteri ci sono due modi:

Il primo si ha utilizzando **%10s** che considera solo i primi dieci caratteri inseriti, bisogna dichiarare la stringa di 11 caratteri perché l'ultimo è il terminatore null.

```
77      scanf("%10s", stringa);
```

Il secondo modo è utilizzare la funzione fgets, bisogna pulire il buffer per far vedere la stringa correttamente. In questo modo verranno considerati solo i primi dieci caratteri.

In entrambi i modi si evita lo stack overflow che può essere sfruttato dai black hat.

```
74 void clearInputBuffer() {
75     int c;
76     while ((c = getchar()) != '\n' && c != EOF);
77 }
78 
79 void ins_string() {
80     char stringa[11];
81 
82     clearInputBuffer();
83     printf("Inserisci la stringa: ");
84     fgets(stringa, sizeof(stringa), stdin);
```

Davide Lecci