EPICODE-CS0124
S2/L5 - Progetto

Flaviano Sedici

Progetto

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.

1. Capire cosa fa il programma

1.1. Descrizione del programma

- Il programma è composto, oltre che dalla funzione main(), anche da altre 4 funzione che vengono chiamate all'interno del codice main().
- Il programma ha come scopo quello di chiedere all'utente la funzione, tra quelle 3 disponibili, che vuole eseguire, ossia la moltiplicazione, la divisione o l'inserimento di una stringa.
- A seguito della scelta effettuata tramite l'immissione delle lettere A, B o C, il programma invoca le funzioni (rispettivamente) moltiplica(), dividi() e ins_string().
- La selezione viene gestita dal programma tramite la funzione switch.
- La funzione moltiplica() richiede all'utente di inserire due numeri, li moltiplica e restituisce una stampa a terminale del risultato.
- La funzione dividi() richiede all'utente di inserire due numeri, li divide e restituisce una stampa a terminale del risultato.
- La funzione ins_string() richiede all'utente di inserire una stringa senza restituire alcun risultato a terminale.
- La funzione menu() effettua semplicemente il printf del messaggio di benvenuto e delle scelte possibili.

1.2. Debug del codice

```
#include <stdio.h>
void menu ();
void moltiplica ();
void dividi ();
void ins_string();
int main ()
{
	char scelta = {'\0'};
```

//Errore di sintassi: la variabile è stata inizializzata come char ma gli è stato assegnato un valore come se fosse un array di caratteri. La scelta migliore è di inizializzare il valore come un carattere singolo perché non è necessario un array, quindi modificherei la sintassi come **char scelta = ‘ ‘;**

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

//Errore di sintassi: se la scelta è inizializzata come array c'è un errore di sintassi perché dovremmo mettere %c al posto di %d.

```
switch (scelta)
{
        case 'A':
        moltiplica();
        break;
        case 'B':
    dividi();
    break;
        case 'C':
    ins_string();
    break;
```

//Comportamento non contemplato: inserirei un **default** in modo da gestire efficacemente input diversi da quelli programmati, con un **printf** che indichi all'utente che non ha inserito una lettera corrispondente ad una funzione esistente riavviando la funzione **menu()**

//Comportamento migliorabile: si potrebbe decidere di accettare tramite l'operatore **|| (OR)** anche le lettere minuscole a, b, c come input corretti, ad esempio **case ‘A’ || ‘a’:**

```
    }
return 0;
}
void menu ()
{
    printf ("Benvenuto, sono un assitente 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);
```

//Errore di sintassi: la %f è errata perché ci aspettiamo uno short int e quindi la notazione corretta è %hd

//Comportamento non contemplato: potremmo inserire un controllo per verificare che quanto inserito sia uno short int e restituire un errore ripetendo la domanda nel caso non lo sia: if(scanf(“ %hd”, &a)!=1) printf(“il numero inserito non è uno short int”);

//Comportamento migliorabile: si potrebbe inizializzare le variabili a, b, e prodotto come come int in modo da limitare eventuali errori di overflow se si dovessero utilizzare interi più lunghi

```
        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 denumeratore:");
        scanf ("%d", &b);
```

//Comportamento non contemplato: potremmo inserire un controllo per verificare che quanto inserito sia uno short int e restituire un errore ripetendo la domanda nel caso non lo sia: if(scanf(“ %hd”, &a)!=1) {printf(“il numero inserito non è uno short int”); return 1}

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

//Errore logico: nella divisione viene utilizzato solitamente il simbolo / e non % che è invece il resto della divisione e non la divisione in sé

//Comportamento migliorabile: divisione andrebbe gestito come float perché non sempre la divisione tra due numeri restituisce un intero. Inoltre andrebbe effettuata la conversione a float di a e b con il comando (float)a / (float)b per far sì che la loro divisione restituisca un float.

//Comportamento non contemplato: sarebbe necessario verificare che il valore acquisito b non sia pari a 0 e nel caso restituire un errore: if(b==0) {printf(“Nn è possibile dividere per 0t”); return 1}

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

//Comportamento migliorabile: dovendo gestire la divisione come float (vedi sopra) il terzo placeholder %d dovrebbe essere sostiuito con un %f o %.2f

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

//Errore di sintassi: nelle stringhe, essendo degli array di caratteri, il puntatore deve essere assegnato al primo valore dell'array in quanto l'array stesso è posizionato su più blocchi di memoria sequenziali che si concludono con il valore

\0. Quindi inserire la & davanti alla stringa nello scanf è errato perché il valore stringa senza la & di per sé restituisce l'indirizzo del primo blocco di memoria da cui parte l'array.

//Comportamento migliorabile: avendo impostato una stringa con massima lunghezza 10, prima di tutto la imposterei tutta ad un valore vuoto = {' '} per evitare di visualizzare i valori casuali presenti nei blocchi di memoria riservati all'array. Poi inserirei un controllo nello scanf con una Regex in modo che prenda solo i primi 10 caratteri per evitare un overflow : ad esempio scanf(" %10s", stringa);

//Comportamento migliorabile: si potrebbe inserire una stampa a terminale del valore stringa inserito solo per coerenza con le altre funzioni

```
}
```