Analisi di un codice in C: riconoscimento errori e possibili soluzioni.

Di seguito verrà analizzato un semplice programma scritto in C: ne verrà chiarito lo scopo, ne saranno individuati diversi errori sia logici che di sintassi e saranno evidenziate le possibili casistiche a cui il programma non prevede, provvedendo una soluzione ad ognuno di essi.

Innanzitutto, il programma presentato è una semplice calcolatrice: offre all'utente la possibilità di moltiplicare o dividere due numeri tra loro, oltre alla possibilità di inserire una breve stringa di testo.

Di seguito, saranno discussi gli errori individuati punto per punto, offrendo per ciascuno di essi una possibile soluzione. Sarà allegato con questo file il codice esaminato (nome file: "Codice errato", da visualizzare a schermo intero per mantenere corretta l'impaginazione) come riferimento, con alcune annotazioni (espresse tramite commento "//") per identificare gli errori qui di seguito riportati e alcune imprecisioni nella sintassi. Inoltre, sarà allegato un altro file di codice ("Codice corretto") contenente una versione corretta del programma per evidenziare ulteriormente le differenze nei due.

Iniziando con l'analisi, qui di seguito vi sono tutti gli errori incontrati:

1. La variabile "**scelta**" viene definita come **char**, di conseguenza l'input in entrata deve essere definito con la sintassi "**%c**" e non "**%d**", utilizzata per variabili numeriche intere di 4 byte.
2. Nel codice preso fin qui in esame, viene chiesto all'utente di inserire un input, assegnato alla variabile "**scelta**", su cui successivamente si basa una funzione switch. All' interno di questa però non è prevista alcuna opzione nel caso in cui l'utente inserisca un input diverso da '**A**', '**B**' o '**C**', portando quindi in questo caso all'arresto del programma. Possiamo ovviare al problema inserendo questo switch in un ciclo while che avverta l'utente della scelta errata e dia la possibilità di inserire un nuovo input, come mostrato nel file "Codice corretto".
3. Le variabili vengono qui definite sia come **short** che come **int**; considerando che nella funzione **printf()** che mostra il risultato dell' operazione abbiamo la sintassi "**%d**" (dedicata a **int**), diamo per assodato che le variabili siano da definire con **int**, permettendo inoltre di avere un maggior range di numeri disponibili (abbiamo infatti quattro byte di memoria dedicata alla variabile invece di due). Seguendo lo stesso ragionamento, alla linea 47 la sintassi corretta è "**%d**" e alla linea 50 ancora definiamo il prodotto come **int**. Volendo andare sul sicuro, se ci aspettiamo che gli input dell'utente possano essere numeri molto grandi, potremmo addirittura definire il prodotto come **long**, assegnandogli ben otto byte di memoria, e cambiando la relativa sintassi alla linea 52 con "**%l**".
4. La funzione presa qui in esame ha lo scopo di svolgere una divisione ma non prende in considerazione la possibilità che l'utente inserisca 0 al denominatore. Possiamo risolvere il problema inserendo un loop while, il che permette anche all'utente di inserire un nuovo valore, come mostrato nel file "Codice corretto".
5. Essendo questo il risultato di una divisione, converrebbe definire la variabile con **float** (modificando quindi anche la linea 66 con "**%f**" nel punto dedicato al risultato), permettendo quindi la presenza di decimali. Inoltre, l'operatore corretto è "/", in quanto "**%**" restituirebbe solo il resto della divisione.
6. L' ultima parte del codice permette all'utente di inserire in input una breve stringa di testo, senza però restituire alcun riscontro. Potremmo quindi inserire un **printf()** relativo all' input appena inserito per mostrare all' utente la stringa appena inserita.
7. Infine, nella definizione della variabile stringa viene specificato che questa deve avere una lunghezza massima di 10 caratteri; possiamo fare ciò inserendo un limite al numero di caratteri che possono essere inseriti in input dall'utente usando la sintassi "**%10s**" nella funzione **scanf().**

I due codici vengono inoltre riportati nelle pagine seguenti, in modo tale da avere tutto il necessario in un unico documento.

Codice errato

```c
#include <stdio.h>

void menu ();
void moltiplica ();
void dividi ();
void ins_string ();


int main ()

{
	char scelta = {'\0'};		// Le parentesi graffe non sono necessarie
	menu ();
	scanf ("%d", &scelta);		// 1) Essendo la variabile "scelta" specificata come char, l'input utente deve essere definito come "%c"

	switch (scelta)
	{
		case 'A':		// 2) A questa instruzione switch manca l'opzione per contemplare possibili input errati da parte dell'utente,
		moltiplica();		//    come l'inserimento di un carattere non previsto nel loop (ad esempio, nel precedente "scanf ()" l'utente
		break;			//    potrebbe digitare "D" e ciò romprebbe immediatamente il programma). Potremmo ovviare al problema anticipando
		case 'B':		//    a tutto questo codice un ciclo while.
		dividi();
		break;
		case 'C':
		ins_string();
		break;
	}

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\n B >> Dividere due numeri\n C >> Inserire una stringa\n");

}


void moltiplica ()
{
	short int  a,b = 0;					// 3) Vengono dichiarati contemporaneamente due tipi di variabili, quando ne serve sono uno (o short o int);
	printf ("Inserisci i due numeri da moltiplicare:");	//    usiamo int in quanto prevede un numero maggiore di variabili (4 byte invece dei 2 di short).
	scanf ("%f", &a);					//    Avendo precedentemente dichiarato che la variabile "a" è un int, la sintassi corretta è "%d".
	scanf ("%d", &b);

	short int prodotto = a * b;				//    Come prima, definiamo la variabile usando solo int.

	printf ("Il prodotto tra %d e %d e': %d", a,b,prodotto);
}


void dividi ()
{							// 4) Qui abbiamo una divisione, non viene però presa in considerazione la possibilità che l'utente inserica
	int  a,b = 0;					//    "0" come denominatore. Anche in questo caso, possiamo risolvere il problema con un while permettendo
	printf ("Inserisci il numeratore:");		//     così all' utente di reinserire i dati.
	scanf ("%d", &a);
	printf ("Inserisci il denumeratore:");
	scanf ("%d", &b);

	int divisione = a % b;				// 5) Essendo una divisione, conviene usare float invece di int. Inoltre l'operatore corretto è "/".

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




void ins_string ()					// 6) La seguente funzione permette all'utente di inserire una stringa in input senza però restituire alcun risultato.
{							//    Potremmo ad esempio inserire una funzione di stampa alla fine per mostrare all'utente ciò che ha scritto:
	char stringa[10];
	printf ("Inserisci la stringa:");		// 7) Infine, nella definizione della variabile stringa viene specificato che questa deve avere una lunghezza massima
	scanf ("%s", &stringa);				//    di 10 caratteri; possiamo assicurare ciò inserenso un limite all'input dell'utente inserendo "%10s% invece di
}							//    "%s" nella funzione "scanf()"
```

Codice corretto

```c
#include <stdio.h>

void menu();
void moltiplica();
void dividi();
void ins_string();


int main () {

    char scelta = '\0';
    menu ();
    scanf(" %c", &scelta);

    while (scelta != 'A' && scelta!= 'B' && scelta != 'C') {

        printf("Scelta non valida, inserire un' opzione tra A, B o C. \n");
        menu();
        scanf(" %c", &scelta);
    }

    switch (scelta) {

        case 'A':
        moltiplica();
        break;

        case 'B':
        dividi();
        break;

        case 'C':
        ins_string();
        break;
    }

    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() {

    int a = 0;
    int b = 0;
    printf("Inserisci i due numeri da moltiplicare: \n");
    scanf(" %d", &a);
    scanf(" %d", &b);

    int prodotto = a * b;      // Potremmo usare long se prevediamo che il risultato sia un numero molto grande.

    printf("Il prodotto tra %d e %d e': %d", a, b, prodotto);    // Se precedentemente abbiamo usato long, inseriamo %l al posto dell' ultimo %d.
}

void dividi() {

    int a, b = 0;

    printf("Inserisci il numeratore: \n");
    scanf(" %d", &a);
    printf("Insericsci il denominatore: \n");
    scanf(" %d", &b);

    while (b == 0) {

        printf("Il denominatore non può essere 0, inserire un nuovo valore: \n");
        scanf(" %d", &b);
    }

    float divisione = (float)a / b;

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

}

void ins_string() {

    char stringa[11];
    printf("Inserisci la stringa: \n");
    scanf(" %10s \n", stringa);
    printf("Hai inserito la stringa: %s", stringa);

}
```