Progetto S2L5

Bug hunting, analisi e correzione di un file C

Gli obiettivi di oggi sono:

- Capire il funzionamento del programma
- Individuare situazioni impreviste non gestite dal programma
- Individuare errori di sintassi / logici
- Proporre una soluzione per ognuno di essi

Il programma è multifunzionale. Propone un menu che rende possibile 3 possibili operazioni:

1) moltiplicare 2 numeri
2) dividere 2 numeri
3)inserire una stringa di testo

Sull'immagine soprastante e nelle slide seguenti verrà presentato il codice sorgente, con tanto di commenti relativi a errori sintattico-logici

```c
#include <stdio.h>

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


int main ()

{
        char scelta = {'\0'};
        menu ();
        scanf ("%d", &scelta);  //tipo di dato errato, %d è per il tipo int, non string %s

        switch (scelta)
        {
                case 'A':
                moltiplica();
                break;
                case 'B':
                dividi();
                break;
                case 'C':
                ins_string();
                break;
        //Manca il default all'interno dello switch, il programma non può gestire input diversi da A,B o C, neanche le loro
varianti minuscole
        }
    //il programma non presenta modo di continuare ad essere eseguito una volta finita una operazione
return 0;
```

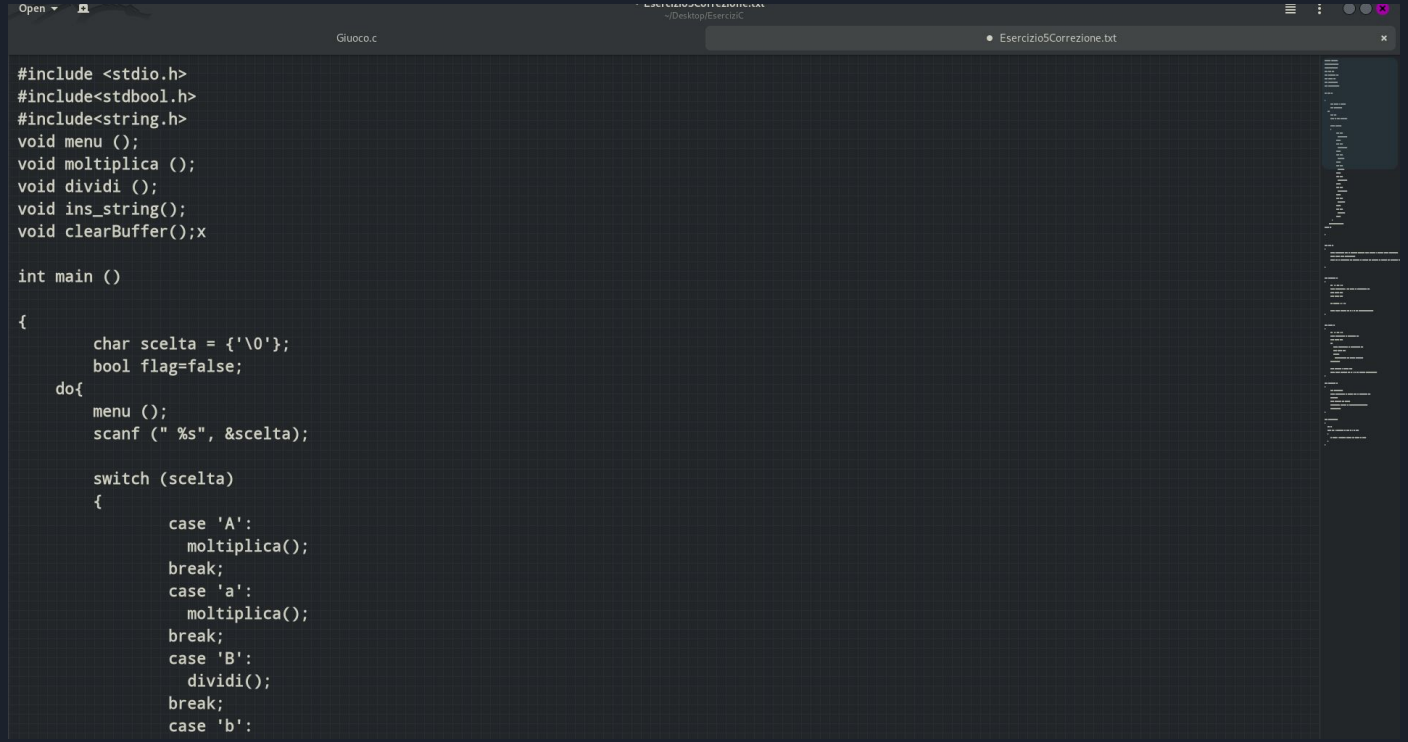
All'interno del main() troviamo diversi errori:
"%d" è utilizzato per il tipo int, ma lo scanf sta ricevendo dati per una stringa.
Non c'è modo di continuare ad usare il programma una volta finita una operazione, oltretutto lo switch è incompleto

```
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);       //tipo di dato errato, %f è per il tipo float, non int %d
        scanf ("%d", &b);

        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:");
```

in moltiplica() il tipo short int è restrittivo, molto semplice da mandare in overflow con una moltiplicazione, e %f raccoglie un file float ma i dati sono short int, appunto.

```c
void dividi ()
{
        int  a,b = 0;
        printf ("Inserisci il numeratore:");
        scanf ("%d", &a);
        printf ("Inserisci il denumeratore:");
        scanf ("%d", &b);       //Nessun controllo per il denominatore, potrebbe essere inserito lo 0

        int divisione = a % b;  //L'operatore utilizzato è il modulo, ma la funzione deve dare il quoziente
                                //La divisione può dare come risultato un valore float, ma l'operazione è svolta in int

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




void ins_string ()
{
        char stringa[10];
        printf ("Inserisci la stringa:");       //Non c'è avviso per l'utente per il massimo dei caratteri inseribili
        scanf ("%s", &stringa);
    //non c'è modo di limitare il numero di caratteri inseriti in input, l'inserimento non ha utilizzo nel programma
}
```

in dividi() non è presente alcun controllo per il denominatore della divisione in caso di inserimento dello 0. L'operazione viene svolta e inserita in una variabile int, che non permette di calcolare la parte frazionaria del risultato. In più l'operando utilizzato è quello del modulo(resto) % e non quello della divisione /



Adesso posso procedere a illustrare la versione corretta del programma, che tiene conto degli errori logici e situazioni tralasciate dal codice originale.

Inizio aggiungendo le librerie “stdbool.h” e “string.h” che utilizzerò in seguito.
modifico il main aggiungendo una variabile bool “flag” che utilizzerò per terminare il programma tramite comando dell’utente. Implemento un do{}while() per fare in modo che il programma ricominci l’esecuzione dopo l’utilizzo di una operazione al suo interno.
Dichiaro la funzione clearBuffer().

```
                case 'b':
                  dividi();
                break;
                case 'C':
                  ins_string();
                break;
                case 'c':
                  ins_string();
                break;
                case 'X':
                  flag=true;
                break;
                case 'x':
                  flag=true;
                break;
            }
        }while(flag==false);
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\nX >> Uscire dal programma\n");

}
```

Ho migliorato lo switch-case, tenendo conto dell'uso delle minuscole.
non c'è bisogno di un caso default dato che in caso di input errato il menu si ripresenterà grazie al do{}while()
Ho aggiunto al printf() del menu l'opzione X per uscire dal programma.

```c
void moltiplica ()
{
        int  a = 0,b = 0;
        printf ("\nInserisci i due numeri da moltiplicare: ");
        scanf ("%d", &a);
        scanf ("%d", &b);

        int prodotto = a * b;

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


void dividi ()
{
        int  a = 0,b = 0;
        printf ("\nInserisci il numeratore: ");
        scanf ("%d", &a);
        do{
            printf ("\nInserisci il denumeratore: ");
            scanf ("%d", &b);
            if(b==0)
              printf("\nValore non valido, riprova");
        }while(b==0);

        float divisione = (float) a/b;
        printf ("\nLa divisione tra %d e %d e': %.3f\n", a,b,divisione);
}
```

Sia in moltiplica() che in dividi() ho inizializzato entrambe le variabili a 0.
ho modificato il tipo delle variabili di moltiplica() ad int.
In dividi() ho implementato un do{}while() nell'inserimento del denominatore e un if() per segnalare l'inserimento errato dello 0 e utilizzato il cast per calcolare e restituire il risultato dell'operazione i

```
            scanf ("%d", &b);
            if(b==0)
              printf("\nValore non valido, riprova");
        }while(b==0);

        float divisione = (float) a/b;
        printf ("\nLa divisione tra %d e %d e': %.2f\n", a,b,divisione);
}

void ins_string ()
{
        char stringa[11];
        printf ("\nInserisci la stringa (max 10 caratteri): ");
        getchar();
        fgets (stringa, 11, stdin);
        printf("\nhai inserito %s correttamente!",stringa);
        clearBuffer();
}

void clearBuffer()
{
    int c;
    while ((c = getchar()) != '\n' && c != EOF)
    {
        // Scarta i caratteri residui nel buffer di input
    }
}
```

Infine, utilizzando getchar(), fgets() e clearBuffer() ho evitato che si potessero inserire più di 10 caratteri (11 considerando il carattere null inserito automaticamente da fgets())