Bug Hunting 2

Cosa fa il codice (Esercizio_10_Epicode.c)

Questo programma è creato per svolgere delle funzioni.

Mostrare un menu principale dove suono riportare le seguenti altre funzioni :

- Moltiplicazioni ;
- Divisioni ;
- Ed inserire una stringa(sequenza di caratteri di ordine prestabilito).

Correzione ed spiegazione del programma

```
#include <stdio.h>
```

È una direttiva del processore, come vedete è preceduta da #. Questa riga indica al processore, che prima di iniziare a leggere le istruzioni del programma, deve caricare in memoria la libreria citata, che contiene le definizioni delle funzioni per l'input/output standard in un programma C.

Dopo di che vengono dichiarate delle funzioni

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

Per esempio la dichiarazione **"void ins_string();**" indica la presenza di una funzione chiamata "ins_string" che restituisce un valore di tipo "void". La parola chiave "void" viene utilizzata per indicare che la funzione non restituisce alcun valore di ritorno. Quando una funzione ha un tipo di ritorno "void", significa che la funzione esegue un'azione o un'operazione senza restituire un risultato specifico.

Dopo viene eseguita la dichiarazione "in name"

```
int main ()

{
      char scelta = {'\0'};
      menu ();
      scanf ("%d", &scelta);

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

return 0;

}
```

La dichiarazione **"int main()"** indica la presenza della funzione principale del programma, denominata "main". La funzione "main" è il punto di ingresso del programma C. È la prima funzione che viene eseguita quando il programma viene avviato. Da cui inizia l'esecuzione del codice.

La parola chiave "int" prima del nome della funzione "main" indica che la funzione restituisce un valore di tipo intero (int). Il valore restituito da "main" rappresenta lo stato di terminazione del programma e viene spesso utilizzato per indicare se il programma è stato eseguito correttamente o se si è verificato un errore

La dichiarazione **"char scelta;**" in C indica la definizione di una variabile di tipo "char" chiamata "scelta". La variabile "scelta" può essere utilizzata per memorizzare un singolo carattere. Non ha bisogno di essere inizializzata = {'\0'}.

Con **menu();** vado a chiamare la funzione menu.

La chiamata **scanf(“%d”)**è sbagliate perché la variabile **scelta** è un carattere e non un intero per cui La chiamata giusta è a chiamata **scanf("%c", &scelta)**; in C indica che si sta cercando di acquisire un singolo carattere dall'input dell'utente e memorizzarlo nella variabile scelta. La specifica del formato %c di scanf viene utilizzata per acquisire un carattere dalla console.

L'operatore & viene utilizzato per ottenere l'indirizzo di memoria della variabile scelta da passare a scanf. In questo modo, scanf leggerà un carattere dalla console e lo assegnerà alla variabile scelta.

```
{
	char scelta;
	menu ();
	scanf ("%c", &scelta);
```

switch (scelta)

L’ istruzione switch in C che gestisce il valore della variabile scelta e esegue diverse azioni in base al suo valore. In particolare, il codice esegue le seguenti operazioni:

Se scelta è uguale a 'A', viene chiamata la **funzione moltiplica()**.

Se scelta è uguale a 'B', viene chiamata la **funzione dividi()**.

Se scelta è uguale a 'C', viene chiamata la **funzione ins_string()**.

Dopo il blocco switch, l'istruzione **return 0**; restituisce un valore di 0 dalla funzione main(), indicando che il programma è terminato correttamente.

Si noti che il codice non include le definizioni delle funzioni moltiplica(), dividi(), ins_string() o la dichiarazione di scelta. Per cui queste funzioni devono essere dichiarare prima delle variabile scelta nel dello switch prima di utilizzarle.

```
int main ()

{
void menu ();
void moltiplica ();
void dividi ();
void ins_string();
      char scelta;
      menu ();
      scanf ("%c", &scelta);

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

return 0;

}
```

Menu

```
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");

}
```

Qui l'errore sta del non separare le opzioni dello switch in più printf per cui

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

}
```

La funzione **menu()**rappresenta un menu di opzioni per l'assistente digitale. La funzione stampa una serie di messaggi a schermo per presentare le opzioni disponibili all'utente. Ecco cosa fa la funzione:

Benvenuto, sono un assistente digitale, posso aiutarti a sbrigare alcuni compiti

Come posso aiutarti?

>> Moltiplicare due numeri

>> Dividere due numeri\n")

C >> Inserire una stringa\n")

La funzione menu() è chiamata per vedere quali opzioni può eseguire la machina.

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

      short int prodotto = a * b;

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

La dichiarazione "**short int a, b = 0**;" in C indica la definizione di due variabili di tipo "short int" chiamate "a" e "b". Inoltre, viene assegnato il valore 0 alla variabile "b" durante la dichiarazione. Cose che non servono per cui.

```
void moltiplica ()
{
      int  a,b;
      printf ("Inserisci i due numeri da moltiplicare:");
      scanf ("%d", &a);
      scanf ("%d", &b);

      int prodotto = a * b;

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

La funzione **moltiplica()** esegue la moltiplicazione di due numeri inseriti dall'utente e stampa il risultato. Ecco cosa fa la funzione:

Viene dichiarata e inizializzata una variabile **int a, b**;. Queste variabili verranno utilizzate per memorizzare i due numeri inseriti dall'utente.

Viene **stampato** il messaggio "Inserisci i due numeri da moltiplicare:" per richiedere all'utente di inserire i due numeri da moltiplicare.

Viene utilizzata la funzione **scanf("%d", &a)**; per acquisire il primo numero inserito dall'utente e memorizzarlo nella variabile a.

Viene utilizzata la funzione **scanf("%d", &b);** per acquisire il secondo numero inserito dall'utente e memorizzarlo nella variabile b.

Viene dichiarata e inizializzata una variabile **int prodotto = a * b**;. Questa variabile memorizza il risultato della moltiplicazione dei due numeri.

Viene stampato il messaggio "Il prodotto **tra %d e %d e': %d"** utilizzando **printf,** in cui **%d** rappresenta i segnaposto per i valori delle variabili a, b e prodotto. In questo modo, viene mostrato il risultato della moltiplicazione.

Dividi

```
void dividi ()
{
       int  a,b = 0;
       printf ("Inserisci il numeratore:");
       scanf ("%d", &a);
     printf ("Inserisci il denumeratore:");
       scanf ("%d", &b);

       int divisione = a % b;

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

L'operato **%** divide due numeri e restituisce come risultato il resto non è ottimale in questo caso visto che vogliamo solo dividere per cui **/** e non serve dare **0 ad a e b** .

```
void dividi ()
{
    int  a,b;
    printf ("Inserisci il numeratore:");
    scanf ("%d", &a);
    printf ("Inserisci il denumeratore:");
    scanf ("%d", &b);

    int divisione = a / b;

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

La funzione **dividi()** esegue la divisione di due numeri inseriti dall'utente e stampa il risultato. Ecco cosa fa la funzione:

Viene dichiarata una **variabile int a** per memorizzare il numeratore e una variabile **int b** per memorizzare il denominatore.

Viene **stampato** il messaggio "Inserisci il numeratore:" per richiedere all'utente di inserire il numeratore.

Viene utilizzata la funzione **scanf("%d", &a);** per acquisire il valore del numeratore inserito dall'utente e memorizzarlo nella variabile a.

Viene stampato il messaggio "Inserisci il denominatore:" per richiedere all'utente di inserire il denominatore.

Viene utilizzata la funzione **scanf("%d", &b);** per acquisire il valore del denominatore inserito dall'utente e memorizzarlo nella variabile b.

Viene dichiarata una variabile **int divisione = a / b;** per memorizzare il risultato della divisione tra il numeratore e il denominatore.

Viene stampato il messaggio "**La divisione tra %d e %d e': %d"** utilizzando **printf**, in cui **%d** rappresenta i segnaposto per i valori delle variabili a, b e divisione. In questo modo, viene mostrato il risultato della divisione.

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

La funzione ins_string()

acquisisce una stringa di massimo 10 caratteri inserita dall'utente. Ecco cosa fa la funzione:

Viene dichiarato un array di caratteri **char stringa[10]**; che può memorizzare una stringa di massimo 10 caratteri.

Viene stampato il messaggio "Inserisci la stringa:" per richiedere all'utente di inserire una stringa.

Viene utilizzata la funzione **scanf("%s", &stringa)**; per acquisire la stringa inserita dall'utente e memorizzarla nell'array stringa.

Tuttavia, c'è un problema nel codice. Quando si utilizza **scanf("%s", &stringa)**;, non è necessario utilizzare l'operatore di indirizzo (&) con l'array stringa. È sufficiente passare il nome dell'array stringa senza l'operatore di indirizzo: **scanf("%s", stringa);**. L'array è già un puntatore al primo elemento, quindi non è necessario utilizzare l'operatore di indirizzo.

Inoltre, è importante notare che l'utilizzo di **scanf("%s", stringa**); potrebbe causare un problema di sicurezza noto come "buffer overflow" se l'utente inserisce una stringa più lunga di 10 caratteri.

Quindi per risolvere

```
void ins_string ()
{
      char stringa[10];
    printf ("Inserisci la stringa:");
    scanf ("%s", stringa);
      if(strlen(stringa)>10)
      {
            printf("la sringa ha + di 10 carateri scrivine una con massimo 10 carateri /n");
            memset(stringa,0,sizeof(stringa));
            scanf("%s", stringa);
      }
}
```

memset(str, 0, sizeof(str)) imposta tutti i byte dell'array str a zero. Successivamente, viene stampato il contenuto dell'array che risulterà essere una stringa vuota.

È importante notare che memset opera a livello di byte, quindi se viene utilizzato per inizializzare un array di interi o di altri tipi di dati più grandi di un byte, il valore specificato verrà copiato nei byte di memoria corrispondenti.

Per utilizzare **scanf ("%s", stringa); ed memset(stringa,0,sizeof(stringa));** abbiamo bisogno della libreria <**string.h>** per cui .

```
#include <stdio.h>
#include <string.h>
```

Codice funzionante

```c
#include <stdio.h>
#include <string.h>


int main ()

{
void menu ();
void moltiplica ();
void dividi ();
void ins_string();
      char scelta;
      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\n");
      printf ("B >> Dividere due numeri\n");
      printf ("C>> Inserire una stringa\n");

}
```

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

      int prodotto = a * b;

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


void dividi ()
{
    int  a,b;
    printf ("Inserisci il numeratore:");
    scanf ("%d", &a);
    printf ("Inserisci il denumeratore:");
    scanf ("%d", &b);

    int divisione = a / b;

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



void ins_string ()
{
      char stringa[10];
    printf ("Inserisci la stringa:");
    scanf ("%s", stringa);
      if(strlen(stringa)>10)
      {
            printf("la sringa ha + di 10 carateri scrivine una con massimo 10 carateri /n");
            memset(stringa,0,sizeof(stringa));
            scanf("%s", stringa);
      }
}
```

Risoluzione e gestione dei problemi nel programma

Divisione nella divisione non si può dividere per 0 ; ed se l'utente inserisce un altro carattere.

```
void dividi ()
{
    int  a,b;
    printf ("Inserisci il numeratore:");
    scanf ("%d", &a);
      printf ("Inserisci il denumeratore:");
    scanf ("%d", &b);

      if (b != 0)
    {
        int divisione = a / b;
        printf("La divisione tra %d e %d è: %d\n", a, b, divisione);
    }
    else
    {
        printf("Impossibile dividere per zero.\n");
    }

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

La funzione **if** e **else** in C sono utilizzate per creare strutture di controllo condizionale all'interno di un programma. Queste strutture consentono di eseguire diverse in blocchi di codice a seconda del valore di una condizione booleana.

Cosa succede se a e b non sono interi si rompe per cui risolvo con **do; while** ed utilizzio le variabili **result_a** e **result_b** per memorizzare il valore restituito dalla funzione scanf(). Se scanf() non riesce a leggere un intero valido, il valore restituito sarà diverso da 1.

```
void dividi()
{
    int a, b;
    int divisione;
    int result_a, result_b;

    do {
        printf("Inserisci il numeratore: ");
        result_a = scanf("%d", &a);

        printf("Inserisci il denominatore: ");
        result_b = scanf("%d", &b);

        // Controllo se l'input ha avuto successo e se l'utente ha inserito numeri validi
        if (result_a != 1 || result_b != 1) {
            printf("Input non valido. Assicurati di inserire due numeri interi. Riprova.\n");
            // svuolto buffer degli input
            while (getchar() != '\n');
        }
        else if (b == 0) {
            printf("Impossibile dividere per zero. Riprova.\n");
        }
    } while (result_a != 1 || result_b != 1 || b == 0);

    divisione = (int)a / b;
    printf("La divisione tra %d e %d è: %.2f\n", a, b, divisione);

}
```

Moltiplica

Ora mettiamo caso che l'utente digiti sbagliato ; semplicemente al posto che un numero intero una lettera , gli inserisco un do while **(scanf("%d", &b) != 1)**; che mi dice se la scanf non trova un numero intero ripeti.

Ed se l'utente non sa che 0 per un numero fa 0 gli vado a dire 0 per un numero fa 0 con **f (prodotto == 0)** il prodotto è uguale a zero allora digli che **0 * 0 è 0** se no **else printf("Il prodotto tra %d e %d è: %d\n", a, b, prodotto);**

```
void moltiplica ()
{
    int a, b;

    do {
        printf("Inserisci il primo numero intero da moltiplicare: ");
    } while (scanf("%d", &a) != 1);

    do {
        printf("Inserisci il secondo numero intero da moltiplicare: ");
    } while (scanf("%d", &b) != 1);

    int prodotto = a * b;

    if (prodotto == 0) {
        printf("Un numero per 0 fa 0.\n");
    } else printf("Il prodotto tra %d e %d è: %d\n", a, b, prodotto);
}
```

Menu

mi sembrava carino dare la possibilità di uscire.

```
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");
      printf ("B >> Dividere due numeri\n");
      printf ("C>> Inserire una stringa\n");
    printf ("D>> Esci");
}
```

Main

default: printf("pultoppo l'istruzione non è valida"); selta = ' ' //resetta la scelta non lavida. In caso l'utente non digitasse correttamente le istruzioni.

getcart(); svuolta chart

```
int main ()

{
void menu ();
void moltiplica ();
void dividi ();
void ins_string();
      char scelta;
      menu ();
      scanf ("%c", &scelta);

      switch (scelta)
      {
            case 'A':
                moltiplica();
                break;
            case 'B':
            dividi();
            break;
            case 'C':
            ins_string();
            break;
            case 'D' :
                return = 0 ;
            default:
                printf("pultoppo l'istruzione non è valida");
                  selta = ' ' //resetta la scelta non lavida.
                  break;
      }
getchar();

}
```

Codice Funzione

```
#include <stdio.h>
#include <string.h>


int main ()

{
void menu ();
void moltiplica ();
void dividi ();
void ins_string();
      char scelta;
      menu ();
      scanf ("%c", &scelta);

      switch (scelta)
      {
            case 'A':
                moltiplica();
                break;
            case 'B':
            dividi();
            break;
            case 'C':
            ins_string();
            break;
            case 'D' :
                return = 0 ;
            default:
                printf("pultoppo l'istruzione non è valida");
                  selta = ' ' //resetta la scelta non lavida.
                  break;
      }
getchar();

}


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");
      printf ("B >> Dividere due numeri\n");
      printf ("C>> Inserire una stringa\n");
      printf ("D>> Esci");
}
```

```
void moltiplica ()
{
    int a, b;

    do {
        printf("Inserisci il primo numero intero da moltiplicare: ");
    } while (scanf("%d", &a) != 1);

    do {
        printf("Inserisci il secondo numero intero da moltiplicare: ");
    } while (scanf("%d", &b) != 1);

    int prodotto = a * b;

    if (prodotto == 0) {printf("Un numero per 0 fa 0.\n");}
    else printf("Il prodotto tra %d e %d è: %d\n", a, b, prodotto);
}



void dividi()
{
    int a, b;
    int divisione;
    int result_a, result_b;

    do {
        printf("Inserisci il numeratore: ");
        result_a = scanf("%d", &a);

        printf("Inserisci il denominatore: ");
        result_b = scanf("%d", &b);

        // Controllo se l'input ha avuto successo e se l'utente ha inserito numeri validi
        if (result_a != 1 || result_b != 1) {
            printf("Input non valido. Assicurati di inserire due numeri interi. Riprova.\n");
            // svuolto buffer degli input
            while (getchar() != '\n');
        }
        else if (b == 0) printf("Impossibile dividere per zero. Riprova.\n");
    } while (result_a != 1 || result_b != 1 || b == 0);

    divisione = (int)a / b;
    printf("La divisione tra %d e %d è: %.2f\n", a, b, divisione);

}
```

```
void ins_string ()
{
    char stringa[10];
    printf ("Inserisci la stringa:");
    scanf ("%s", stringa);
      if(strlen(stringa)>10)
      {
            printf("la sringa ha + di 10 carateri scrivine una con massimo 10 carateri /n");
            memset(stringa,0,sizeof(stringa));
            scanf("%s", stringa);
      }
}
```

.