PROGETTO S2/L5: ANALISI E CORREZIONE DEL CODICE IN LINGUAGGIO C

GNU nano 7.2 S2L5.c *

```c
#include <stdio.h>

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


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;

}


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);

        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);

        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);
}
```

Analizzando il codice vediamo alcune problematiche nel codice, che possono generare comportamenti indesiderati, oltre a diversi errori tra cui errori di sintassi e di logica che vediamo in seguito.

Le parentesi graffe non servono, basta dichiarare 'char scelta; '

La variabile è tipo char, ma qui si cerca di leggere un intero usando '%d', va corretto in '%c'

Per rendere il testo più organizzato e leggibile, alla fine delle stringhe di output, aggiungiamo '\n' per ottenere una formattazione migliore .

L'utilizzo di 'short int' non è la scelta ottimale per vari motivi tra cui, possibili problemi di prestazioni, avendo una dimensione di 2 byte, crea dei vincoli critici di memoria, a causa di spazio insufficiente alcune informazioni potrebbero essere presentate in modo incorretto.
Non serve dichiarare le variabili uguale a 0, perché il valore lo definisce l'utente ;

Le variabili sono dichiarate come 'short int', ma vengono poi lette come 'float' e 'int', anche il risultato viene dichiarato come 'short int', va corretto tutto in 'int' per poter gestire il risultato in modo corretto.

Non serve dichiarare le variabili uguale a 0, perché il valore lo definisce l'utente ;

Inoltre sarebbe opportuno utilizzare una variabile di tipo 'float' anziché 'int' per eventuali risultati con i numeri decimali.

L'operatore da utilizzare per la divisione è '/' , quindi va corretto a posto di '%'.

Aggiungiamo una condizione 'if' 'else', per evitare l'errore di una divisione per un valore di numero 0, che potrebbe creare errori e conflitti.

È già un array e non richiede l'operatore di indirizzo, quindi va rimosso '&';

Inoltre un array di 10 caratteri limita la possibilità di inserimento.

Aggiungiamo "Printf ("%s", stringa);" per visualizzare la stringa scritta dall'utente

^G Help ^O Write Out ^W Where Is ^K Cut ^T Execute ^C Location M-U Undo M-A Set
^X Exit ^R Read File ^\ Replace ^U Paste ^J Justify ^/ Go To Line M-E Redo M-6 Cop

Analizzando il codice possiamo capire che si tratta di un semplice programma in linguaggio C , che ci offre attraverso un menù tre possibili scelte per proseguire: La moltiplicazione di due numeri, la divisione di due numeri e l'inserimento di una stringa. C'erano alcuni errori di sintassi e logica che ho provveduto a segnalare nella pagina precedente e a correggere nel codice qui sotto. Tuttavia, mancano alcuni dettagli da aggiungere per rendere il codice più efficiente a livello di input utente, come la gestione di alcuni bug nel codice che potrebbero essere sfruttati per attaccare l'applicazione.

GNU nano 7.2 giorgio.c

```c
#include <stdio.h>

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

int main()
{
    char scelta;           // Tolto: {'\0'}
    menu();
    scanf(" %c", &scelta); // Sostituito '%d' con '%c' perchè la variabile è di tipo char e non int.
                           // Aggiunto uno spazio prima di %c per ignorare eventuali spazi bianchi

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

    return 0;
}

void menu()
{
    printf("Benvenuto, sono un assistente 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, b;             // Cambiata la variabile da 'short int' a 'int'
    printf("Inserisci i due numeri da moltiplicare:");
    scanf("%d %d", &a, &b);

    int prodotto = a * b; // Cambiata la variabile da 'short int' a 'int'

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

void dividi()
{
    float a, b;
    printf("Inserisci il numeratore:");   // Cambiata la variabile da 'int' a 'float'
    scanf("%f", &a);                      // per eventuale utilizzo di numeri decimali.
    printf("Inserisci il denominatore:");
    scanf("%f", &b);

    if (b != 0)          // Aggiunta della condizione if/else per ulteriori controlli sulla divisione.
    {
        float divisione = (float)a / b;
        printf("La divisione tra %f e %f e': %f\n", a, b, divisione);
    }
    else
    {
        printf("Impossibile dividere per zero.\n");
    }
}

void ins_string()
{
    char stringa[1000];
    printf("Inserisci la stringa:");
    scanf("%s", stringa);
    printf("%s", stringa);     // Aggiunta il comando per stampare la stringa inserita dall'utente.
}
```