

Progetto settimanale epicode (Settimana 2)

Data 01/12/2023

Guglielmo Carratello

L'esercizio di oggi ha lo scopo di allenare l'osservazione critica. 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

Abbiamo visto il codice e commentato i vari errori, di seguito gli screen:



```
GNU nano 7.2          esercizio2.c *
#include <stdio.h>

//Dato che nel programma è presente l'operazione con le stringhe è corretto inserire la libreria apposita per poter fixare i possibili errori della scelta C
// Il programma è composto da un menu principale dove è possibile scegliere il tipo di operazione che
// si desidera fare. Presenta vari problemi di sintassi e bug di funzionamento.

void menu ();
void multiplica ();
void dividi ();
void ins_string();

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

    // Errore bisogna sostituire %d con %c dato che la variabile scelta è un carattere come tipo di dato

    switch (scelta) {
        // Bisogna fixare il menu principale ed evitare possibili errore di esecuzione, sarebbe corretto inserire
        // la funzione switch all'interno di un ciclo do-while al quale inserire anche la variabile menu() con il
        // rispettivo 'scanf' dove "%d" andrà modificato. Questo permette una corretta esecuzione del menu principale ed
        // eventuali problemi di immissione sbagliata del valore scelta
        case 'A':
            multiplica();
            break;
        case 'B':
            dividi();
            break;
        case 'C':
            ins_string();
            break;

        //Qui bisogna aggiungere un caso D, per avere la possibilità di uscire dal programma
        //bisogna aggiungere il default del ciclo, per riavere la possibilità di scelta nel caso si sbagli
        //a digitare la scelta.
    }

    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"); // Qui bisogna aggiungere la scelta D >> esci dal programma
}
```

Il codice sottoposto a revisione è stato sottoposto a un'analisi dettagliata al fine di identificare e correggere eventuali errori di sintassi e di migliorare la struttura complessiva. Di seguito, vengono esposti le correzioni necessarie di sintassi ed i miglioramenti che bisognerebbe apportare:

Libreria per le Stringhe:

Si suggerisce l'aggiunta della libreria `<string.h>`, e tale suggerimento potrebbe essere particolarmente valido se si desidera implementare comandi di sicurezza per garantire la corretta gestione dell'operazione di inserimento di una stringa. L'aggiunta di questa libreria permette l'utilizzo di funzioni di manipolazione delle stringhe, come ' `strlen` ', utile per eseguire controlli aggiuntivi sulla lunghezza della stringa, migliorando così la possibilità presenza di bug nel caso si digita una stringa avente più carattere di quelli specificati.

Tipo di Dato per la Scelta:

Un'importante correzione deve essere apportata sostituendo il formato `%d` con `%c` nella funzione `scanf` relativa alla variabile scelta nel main. Questa correzione è cruciale per assicurare che venga acquisito un carattere anziché un numero intero, contribuendo alla corretta gestione delle scelte dell'utente.

Ciclo do-while e Default del Switch:

Una modifica significativa può essere introdotta incorporando la funzione `switch` all'interno di un ciclo `do-while`. Questa implementazione permette agli utenti di effettuare scelte multiple senza la necessità di riavviare il programma, migliorando così l'usabilità. Inoltre, sono stati aggiunti un caso per l'opzione 'D' per uscire dal programma e un caso di default, offrendo una gestione più completa delle scelte utente ed evitando possibili bug.

Carattere `\n` newline nella dichiarazione della funzione `void menu()`

Una modifica obbligatoria sta nell'inserire il carattere speciale ' `\n` ' al termine della prima stringa

Del comando `printf` della funzione `void menu()`. Questo permette di evitare che le stringhe successive si accostino ad essa durante l'esecuzione del programma.

```
Applications Places Terminal
Dec 1 17:02
kai@kali: ~/Desktop
esercizio2.c *
GNU nano 7.2
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"); // Qui bisogna aggiungere la scelta D >> esci dal programma
}

void moltiplica ()
{
    short int a,b = 0; // è sbagliato scrivere int a,b = 0 perchè scritto così considera come valore iniziale 0 solo la variabile B
    printf ("Inserisci i due numeri da moltiplicare:"); // Bisogna aggiungere il carattere di newline \n a fine stringa
    scanf ("%f", &a); // Qui bisogna sostituire %f con %d dato che si tratta di un dato di tipo intero e non float
    scanf ("%d", &b);

    short int prodotto = a * b; //Sarebbe più corretto aggiungere un ciclo do-while per fixare eventuali problemi di inserimento dati come lettere al posto di numeri

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

void dividi ()
{
    int a,b = 0; // è sbagliato scrivere int a,b = 0 perchè scritto così considera come valore iniziale 0 solo la variabile B
    printf ("Inserisci il numeratore:"); // bisogna aggiungere il carattere newline \n a fine stringa
    scanf ("%d", &a);
    printf ("Inserisci il denominatore:"); // bisogna aggiungere il carattere newline \n a fine stringa
    scanf ("%d", &b);

    int divisione = a % b; // sarebbe più corretto aggiungere un ciclo do-while per fixare eventuali problemi di inserimento dati come lettere al posto di numeri

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

void ins_string ()
{
    char stringa[10]; // Qui dato che il codice specifica il numero massimo di caratteri della variabile stringa, dobbiamo aggiungere comandi
    printf ("Inserisci la stringa:"); // per il quale specifica la lunghezza massima e nel caso si inseriscano numeri il programma ti dia la possibilità di riprovarci
    scanf ("%s", &stringa); // bisogna aggiungere il carattere newline a fine stringa
}

Help Write Out Where Is Cut Execute Location M-U Undo M-A Set Mark M-T To Bracket M-P Previous M-B Back M-V Prev Word M-H Home
Exit Read File Replace Paste Justify Go To Line M-E Redo M-D Copy M-W Where Was M-N Next M-F Forward M-X Next Word M-End End
```

Valore Iniziale di Variabili:

Abbiamo notato un errore nell' inizializzazione delle variabili a e b nelle funzioni moltiplica e dividi. Per come è stato scritto Il problema è che solo b viene inizializzato a 0. Inizializzare entrambe le variabili è cruciale per garantire un comportamento prevedibile durante le operazioni aritmetiche.

Conversione %f in scanf in moltiplica:


La scanf all'interno della funzione moltiplica deve essere corretto sostituendo %f con %d. Questo adeguamento è stato fatto in considerazione del fatto che a e b sono dichiarati come short int, pertanto dovrebbero essere acquisiti come interi e non come float.

Cicli do-while in moltiplica e dividi:

Un'ulteriore proposta è stata avanzata, suggerendo l'implementazione di cicli do-while all'interno delle funzioni moltiplica, dividi e inserisci stringa. L'inserimento di questi cicli potrebbe migliorare significativamente il funzionamento del programma, consentendo agli utenti di correggere eventuali errori di inserimento e prevenendo potenziali bug derivanti da digitazioni errate.

Codice con applicate le correzioni e i possibili miglioramenti

Di seguito, viene fornita l'illustrazione e la spiegazione sulle modifiche apportate:



```
GNU nano 7.2 esercizioepicode.c *
#include <stdio.h>
#include <string.h>

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\nD >> Esci\n");
}

void multiplica ()
{
    int a = 0;
    int b = 0;          // Qui abbiamo sostituito short int con int
    int valore_corretto1;
    int valore_corretto2;
    printf ("Inserisci i due numeri da moltiplicare:\n");

    do {
        printf("Inserisci il primo numero:\n");
        valore_corretto1 = scanf ("%d", &a);

        if(!valore_corretto1) {
            printf("Puoi inserire solo numeri. Riprova.\n");
            while (getchar() != '\n');
        }
    } while(!valore_corretto1);

    do {
        printf ("Inserisci il secondo numero:\n");
        valore_corretto2 = scanf ("%d", &b);
        if(!valore_corretto2){
            printf("Puoi inserire solo numeri. Riprova");
            while (getchar() != '\n');
        }
    }while(!valore_corretto2);

    int prodotto = a * b;    // Qui abbiamo sostituito short int con int

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

GNU nano 7.2 editor interface showing the code. The bottom status bar displays various keyboard shortcuts: ^G Help, ^X Exit, ^O Write Out, ^R Read File, ^W Where Is, ^N Replace, ^K Cut, ^U Paste, ^T Execute, ^J Justify, ^C Location, ^_ Go To Line, M-U Undo, M-E Redo, M-A Set Mark, M-6 Copy, M-J To Bracket, M-O Where Was.

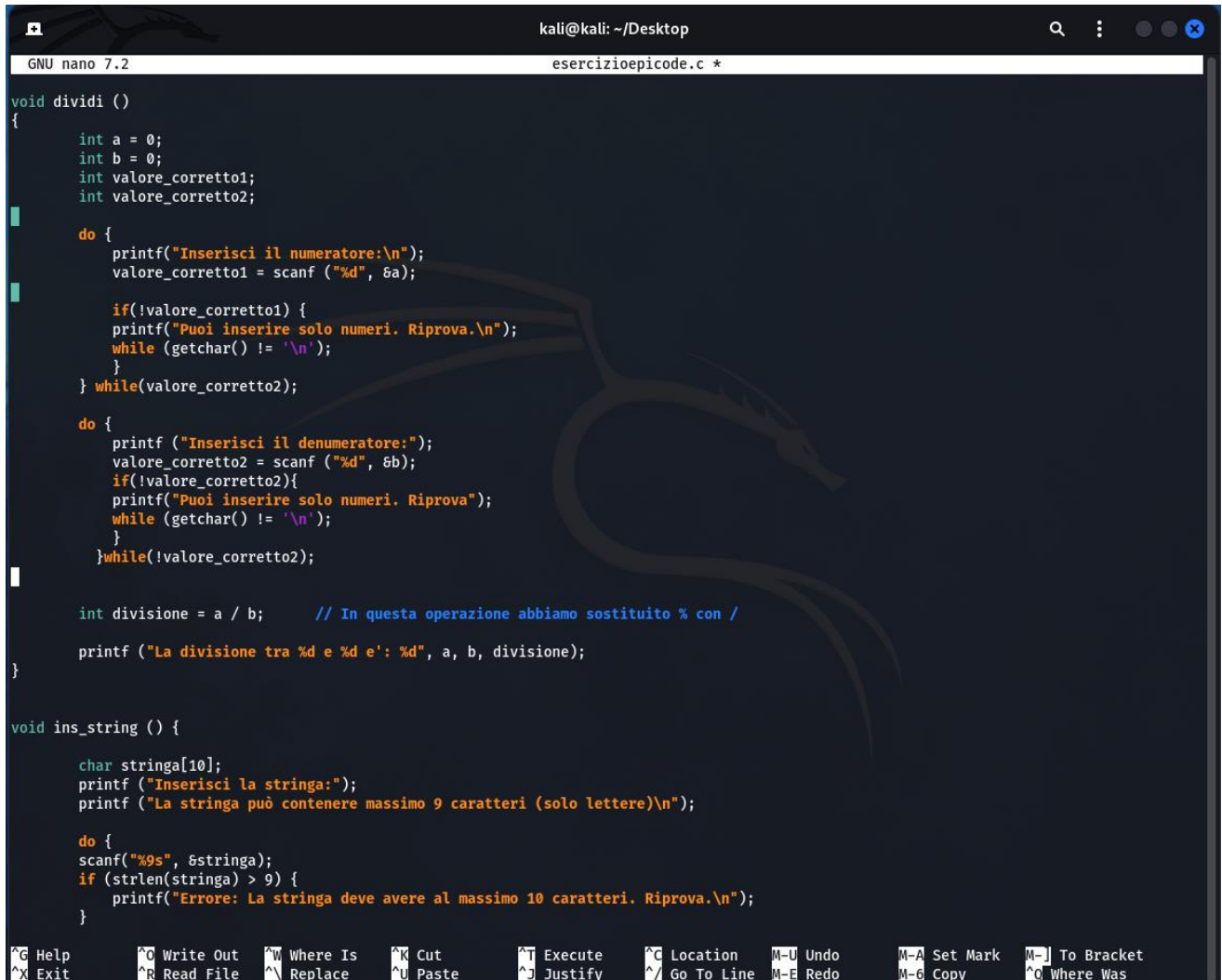
Menu (`void menu()`):

La funzione menu è stata mantenuta essenzialmente invariata. Aggiornato l'output per includere la nuova opzione 'D' per uscire dal programma.

Funzione Moltiplicazione (`void multiplica()`):

Le variabili a e b sono state dichiarate come **int** per garantire che possano contenere numeri interi senza problemi di overflow. È stata aggiunta una variabile **valore_corretto** per controllare se l'input dell'utente è corretto.

L'input dell'utente è ora gestito in due cicli **do-while** separati, uno per ciascun numero da moltiplicare. Questo consente di controllare l'input dell'utente e di chiedere di riprovare in caso di input non valido.



```
void dividi ()
{
    int a = 0;
    int b = 0;
    int valore_corretto1;
    int valore_corretto2;

    do {
        printf("Inserisci il numeratore:\n");
        valore_corretto1 = scanf ("%d", &a);

        if(!valore_corretto1) {
            printf("Puoi inserire solo numeri. Riprova.\n");
            while (getchar() != '\n');
        }
    } while(valore_corretto1 != 1);

    do {
        printf ("Inserisci il denominatore:");
        valore_corretto2 = scanf ("%d", &b);
        if(!valore_corretto2){
            printf("Puoi inserire solo numeri. Riprova");
            while (getchar() != '\n');
        }
    }while(valore_corretto2 != 1);

    int divisione = a / b;    // In questa operazione abbiamo sostituito % con /

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

void ins_string () {
    char stringa[10];
    printf ("Inserisci la stringa:");
    printf ("La stringa può contenere massimo 9 caratteri (solo lettere)\n");

    do {
        scanf("%9s", &stringa);
        if (strlen(stringa) > 9) {
            printf("Errore: La stringa deve avere al massimo 10 caratteri. Riprova.\n");
        }
    }
}
```

Funzione Divisione (**void dividi()**):

La funzione è stata corretta per gestire correttamente la verifica del valore corretto per entrambi i numeri (numeratore e denominatore) da inserire.

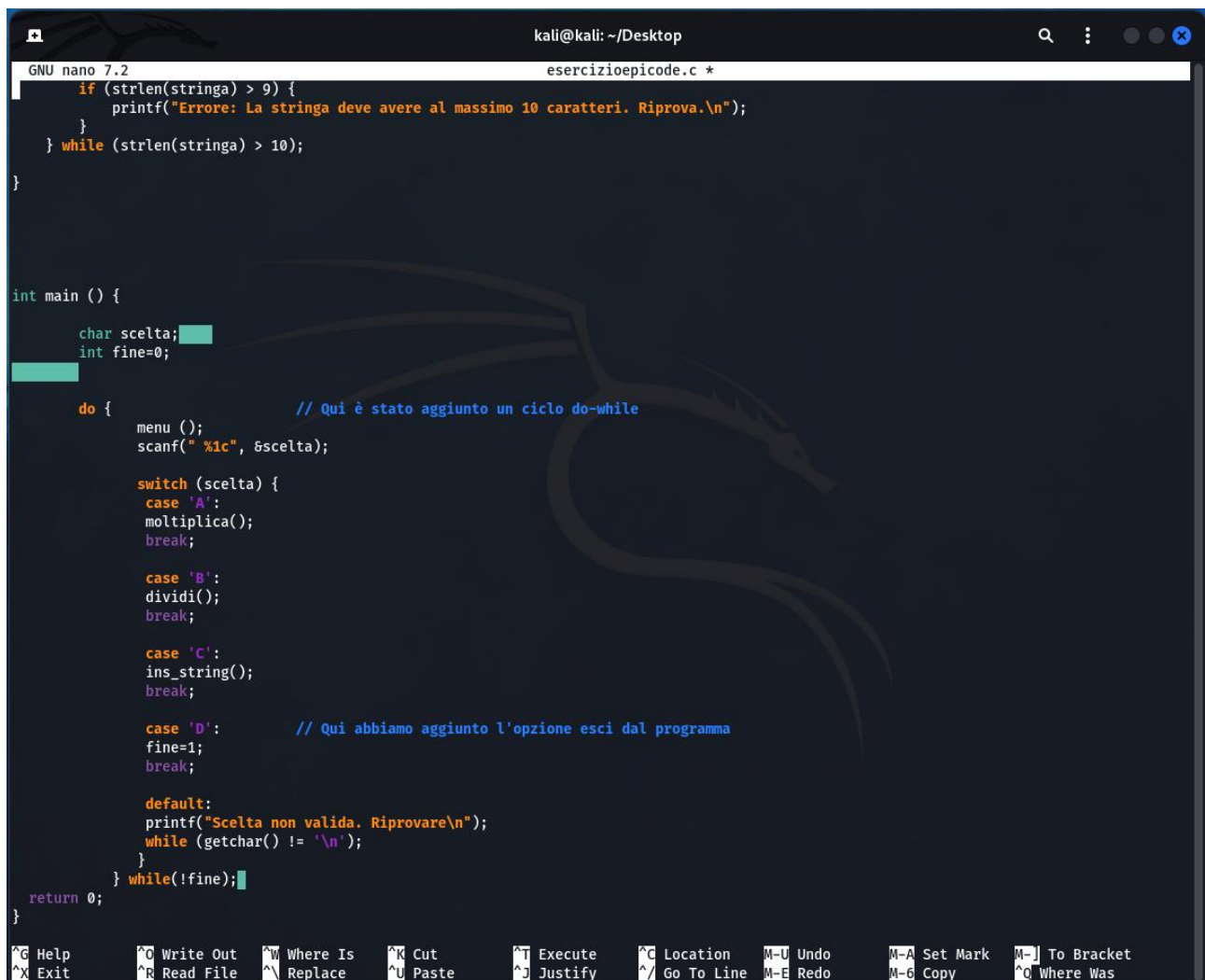
Anche in questo caso, vengono richiesti all'utente nuovi input in caso di inserimenti non validi.

L'operazione di divisione è stata corretta sostituendo % con /.

Funzione Inserimento Stringa (`void ins_string()`):

La variabile stringa è stata dichiarata come `char stringa[10]` per consentire una stringa di lunghezza massima di 9 caratteri, con il decimo riservato per il terminatore nullo (`'\0'`). Abbiamo scritto una stringa facendo presente all'utente che la stringa può contenere solo lettere e non numeri.

È stato aggiunto un ciclo `do-while` per controllare la lunghezza della stringa e richiedere all'utente di riprovare in caso di inserimento di una stringa con più di 9 caratteri. Questo grazie alla funzione `'strlen'` presente nella libreria `string.h`.



```
GNU nano 7.2 esercizioepicode.c *
    if (strlen(stringa) > 9) {
        printf("Errore: La stringa deve avere al massimo 10 caratteri. Riprova.\n");
    }
    } while (strlen(stringa) > 10);
}

int main () {
    char scelta;
    int fine=0;

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

        switch (scelta) {
            case 'A':
                multiplica();
                break;

            case 'B':
                dividi();
                break;

            case 'C':
                ins_string();
                break;

            case 'D':
                // Qui abbiamo aggiunto l'opzione esci dal programma
                fine=1;
                break;

            default:
                printf("Scelta non valida. Riprovare\n");
                while (getchar() != '\n');
        } while(!fine);

    return 0;
}

^G Help      ^O Write Out ^W Where Is  ^K Cut       ^T Execute  ^C Location  M-U Undo    M-A Set Mark M-J To Bracket
^X Exit      ^R Read File ^N Replace   ^U Paste     ^J Justify  ^/_ Go To Line M-E Redo    M-B Copy     ^Q Where Was
```

Funzione Divisione (`void dividi()`):

La funzione è stata corretta per gestire correttamente la verifica del valore corretto per entrambi i numeri (numeratore e denominatore) da inserire.

Anche in questo caso, vengono richiesti all'utente nuovi input in caso di inserimenti non validi.

L'operazione di divisione è stata corretta sostituendo % con /.

Funzione Inserimento Stringa (`void ins_string()`):

La variabile stringa è stata dichiarata come `char stringa[10]` per consentire una stringa di lunghezza massima di 9 caratteri, con il decimo riservato per il terminatore nullo (`'\0'`). Abbiamo scritto una stringa facendo presente all'utente che la stringa può contenere solo lettere e non numeri.

È stato aggiunto un ciclo do-while per controllare la lunghezza della stringa e richiedere all'utente di riprovare in caso di inserimento di una stringa con più di 9 caratteri. Questo grazie alla funzione `'strlen'` presente nella libreria `string.h`.

Menu Principale (`int main()`):

La variabile scelta è stata dichiarata come `char` per rappresentare la scelta dell'utente.

Aggiornato l'input della scelta con `scanf("%1c", &scelta)` per evitare problemi di buffer e consentire una corretta acquisizione del carattere.

Aggiunto il caso 'D' per uscire dal programma nel blocco switch.

Aggiornato il ciclo do-while principale per consentire all'utente di riprovare in caso di scelta non valida, utilizzando un ciclo while (`getchar() != '\n'`) per pulire il buffer di input.

Commenti personali:

Ho scelto di inserire le dichiarazioni delle funzioni void prima della funzione main per garantire una maggiore leggibilità e organizzazione del codice. Questa pratica mi ha facilitato la comprensione del flusso di esecuzione del programma, poiché chi legge il codice può anticipare le operazioni che saranno disponibili nel menu principale.

Ho adottato la medesima metodologia nella scrittura del codice del gioco/quiz della giornata precedente, dimostrandosi efficace nel favorire la comprensione del suo funzionamento.

Conclusioni:

Queste modifiche migliorano la gestione dell'input utente, consentendo il controllo delle operazioni di moltiplicazione, divisione e inserimento di stringhe in modo più robusto e intuitivo, evitando la possibilità che si presentino bug durante l'esecuzione del programma.

