



Progetto S2-L5

Analisi e risoluzione problemi del codice in C

Traccia

Per agire come un Hacker bisogna capire come pensare fuori dagli schemi. 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



Prima parte del codice

Una volta aperto il documento, ho fatto un'analisi sul codice leggendo e annotando tutti gli eventuali errori del programma.

/* In azzurro ho direttamente commentato il codice segnalando eventuali errori e proponendo soluzioni al fine di correggere e migliorare l'esecuzione del programma*/



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

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

int main ()
{
    char scelta = {'\0'}; // non sono necessarie le parentesi graffe
    menu ();
    scanf ("%d", &scelta); // qui andrebbe inserito %c poiché il tipo di scelta è char

    /* dato che il C differenzia le maiuscole dalle minuscole sarebbe meglio mettere dei case aggiuntivi anche
    per a,b,c e inserire caso D,d per uscire dal programma. Inoltre implementerei un messaggio di errore per chiedere di nuovo
    la selezione nel caso in cui l'utente inserisca un carattere non valido e implementerei un ciclo do-while per dare all'utente
    la possibilità di scegliere se fare un'altra operazione o terminare il programma */

    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"); // piccolo errore ortografico nella parola "assistente"
    printf ("Come posso aiutarti?\n");
    printf ("A >> Moltiplicare due numeri\nB >> Dividere due numeri\nC >> Inserire una stringa\n");
}

void moltiplica ()
{
    /* dal momento in cui l'utente non sa che con short int possono essere inseriti solo numeri interi, o inseriamo un messaggio
```



```

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

Seconda parte del codice

Nel codice ho riscontrato sia errori logici, sia di sintassi e addirittura anche ortografici. Ho riportato anche in quest'immagine tutte le mie considerazioni direttamente con un commento nel file di testo.



```
GNU nano 7.2                                progetto.c *

void moltiplica ()
{
    /* dal momento in cui l'utente non sa che con short int possono essere inseriti solo numeri interi, o inseriamo un messaggio
    che lo avverte di questo o meglio cambiare il tipo short int con tipo float per le operazioni tra numeri reali */
    short int a,b = 0;
    printf ("Inserisci i due numeri da moltiplicare:");
    scanf ("%f", &a);
    scanf ("%d", &b); // sostituire con %f

    short int prodotto = a * b; // anche qui sostituire con tipo float

    printf ("Il prodotto tra %d e %d e': %d", a,b,prodotto); // anche qui sostituire con %f ogni %d
                                                                // con %.2f limitiamo a 2 le cifre dopo la virgola per non allungare troppo il numero
}

void dividi ()
{
    int a,b = 0; // anche qui sarebbe meglio mettere tipo float per dividere anche numeri reali
    printf ("Inserisci il numeratore:");
    scanf ("%d", &a); // sostituire con %f
    printf ("Inserisci il denominatore:");
    scanf ("%d", &b); // sostituire con %f

    int divisione = a % b; // per la divisione va inserito / poiché % restituisce solo il resto e cambiare la variabile divisione in tipo float

    printf ("La divisione tra %d e %d e': %d", a,b,divisione); // sostituire con %.2f
}

void ins_string ()
{
    /* dato che la dimensione dell'array è solo di 10 caratteri, possono riscontrarsi problemi nel caso in cui
    l'utente vada ad inserire più di 9 caratteri (nel array uno è riservato per il carattere terminatore null \0)
    per questo è consigliato allungare la dimensione dell'array o specificare all'utente con un printf che si possono
    inserire solo (dimensione dell'array - 1) caratteri */

    char stringa[10];
    printf ("Inserisci la stringa:");
    scanf ("%s", &stringa); /* non c'è bisogno di inserire & poiché un array è già un indirizzo di memoria, inoltre
    inserendo %s per far leggere in input solo i primi 9 caratteri inseriti
    inserirei anche un printf per confermare quello che è stato inserito */
}

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


A questo punto ho applicato
direttamente sul codice le
soluzioni a cui ho pensato



Prima parte del codice

Così come ho fatto per la fase di analisi, ho fatto per la fase di miglioramento, ovvero ho commentato tutte le modifiche direttamente sul file in azzurro. Qui si può notare tutto il corpo modificato della funzione main che adesso ha un ciclo e quindi l'utente può ripetere le operazioni fintanto che desidera farlo.



```
GNU nano 7.2                                progetto2.c *
#include <stdio.h>

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

int main() {
    char scelta = '\0';    // ho rimosso le parentesi graffe

    /* ho inserito un ciclo do-while in modo di darà la possibilità all'utente di scegliere se procedere
    con un'altra operazione oppure terminare il programma al termine di ogni operazione. Ho inoltre
    implementato anche le minuscole, un caso default e il caso D,d per terminare il programma */
    do {
        menu();
        scanf("%c", &scelta);    // ho cambiato con "%c"

        switch (scelta) {
            case 'A':
                moltiplica();
                break;
            case 'a':
                moltiplica();
                break;
            case 'B':
                dividi();
                break;
            case 'b':
                dividi();
                break;
            case 'C':
                ins_string();
                break;
            case 'c':
                ins_string();
                break;
            case 'D':
                printf("Arrivederci!\n");
                break;
            case 'd':
                printf("Arrivederci!\n");
                break;
            default:
                printf("Scelta non valida. Riprova.\n");
        }
    } while (scelta != 'D' && scelta != 'd');    // il ciclo si conclude se l'utente inserisce D,d

    return 0;
}
```

⌘ Help ⌘ Write Out ⌘ Where Is ⌘ Cut ⌘ Execute ⌘ Location ⌘ Undo ⌘ Set Mark ⌘ To Bracket
⌘ Exit ⌘ Read File ⌘ Replace ⌘ Paste ⌘ Justify ⌘ Go To Line ⌘ Redo ⌘ Copy ⌘ Where Was

Seconda e terza parte del codice

Qui mi sono occupato di modificare eventuali errori ortografici, modifiche ai tipi delle variabile dedicate alle operazioni e miglioramento della funzione per la stringa che avrebbe creato grossi problemi di esecuzione se non fossi intervenuto. Inoltre ho aggiunto il caso di errore se si prova a dividere per 0 e un printf di conferma avvenuta immissione della stringa



```
GNU nano 7.2                                progetto2.c *
// correzione della parola "assistente"
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 >> Uscire\n");
}

// come indicato ho modificato i valori affinché si possano svolgere operazioni con numeri reali
void moltiplica() {
    float a, b;
    printf("Inserisci i due numeri da moltiplicare: ");
    scanf("%f%f", &a, &b);

    float prodotto = a * b;

    printf("Il prodotto tra %.2f e %.2f e': %.2f\n", a, b, prodotto);
}

/* stessa cosa per quanto riguarda la divisione e ovviamente modifica di % con /
ho anche inserito un messaggio di errore se si prova a dividere per 0 */
void dividi() {
    float a, b;
    printf("Inserisci il numeratore: ");
    scanf("%f", &a);
    printf("Inserisci il denominatore: ");
    scanf("%f", &b);

    if (b != 0) {
        float divisione = a / b;
        printf("La divisione tra %.2f e %.2f e': %.2f\n", a, b, divisione);
    } else {
        printf("Errore: divisione per zero!\n");
    }
}

/* ho lasciato la dimensione massima della stringa a 10 ma ho fatto comparire un messaggio di avviso
del limite massimo consentito di caratteri all'utente. In questo modo se si immettono più di 9
caratteri il programma ignora i caratteri oltre il nono. Ho leggermente migliorato la funzione
inserendo un piccolo ciclo che mi permette di pulire il buffer di input ed evitare che i caratteri
in eccesso vadano in input non appena compare di nuovo il menù. Ho inserito un printf per far
visualizzare a schermo la stringa inserita fungendo da conferma di corretta esecuzione*/
void ins_string() {
    char stringa[10];
    printf("Inserisci la stringa di massimo 9 caratteri: ");
    scanf("%9s", stringa);

    int c;
    while ((c = getchar()) != '\n' && c != EOF);

    printf("Ecco la stringa che hai inserito: %s\n", stringa);
}

int c;
while ((c = getchar()) != '\n' && c != EOF);

printf("Ecco la stringa che hai inserito: %s\n", stringa);
}
```

Per questo progetto è tutto, tanti saluti



Amedeo Natalizi