Progetto Epicode 06.10.2023

- Comprensione, rilevazione degli errori e ottimizzazione del programma dato

Comprensione

Il programma in linguaggio C, permette di poter scegliere tra 3 opzione (A,B o C), le quali rimanderanno ad una operazione specifica:

- A) moltiplicazione tra due numeri interi, con tetto massimo di digitazione 2^{16} caratteri (da 0 a 32767)

- B) divisione tra due numeri interi con risultato intero (tetto di digitazione 2^{32})

- C) immissione di una stringa di massimo 10 caratteri

Rilevazione degli errori

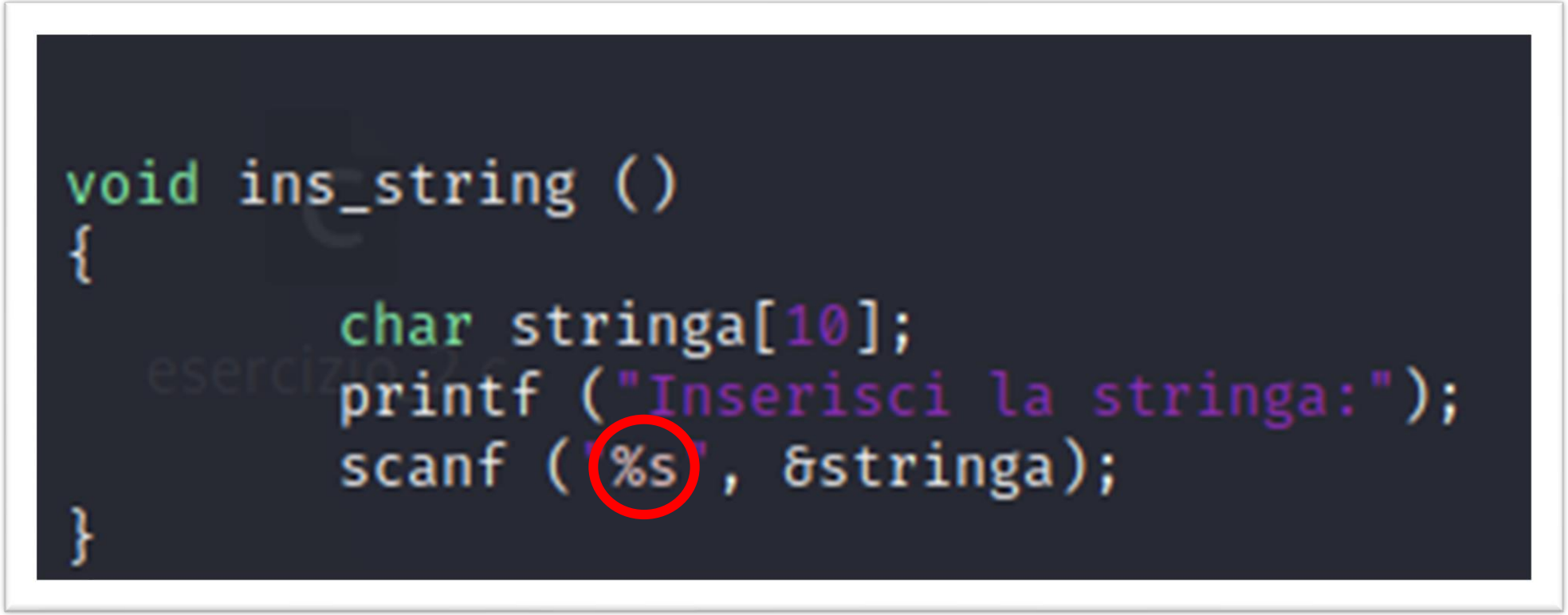
Nella seguente parte del programma (void moltiplica()), troveremo vari errori di sintassi (sono segnati) che dovremmo correggere.

```
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);
}
```

SHORT INT indica il tipo di numero che possono avere le variabili **a** e **b**, in questo caso dei numeri interi (tetto massimo 16 bit).

Il programma una volta eseguito darà errore riguardo i caratteri speciali che servono per l'input e l'output dei numeri che andremmo a scegliere per l'operazione.

Perciò dovremmo cambiare tutti i caratteri speciali segnati con %hd.



In quest'altra parte del programma è presente la variabile char stringa [10], cioè un array con un tetto massimo di 10 caratteri.
Per evitare un eventuale problema di overflow è meglio mettere al posto di «**%s**», come carattere speciale nello **scanf** , un «**%9s**» che leggerà solo i primi 9 caratteri digitati, escludendo a priori gli altri.

Ottimizzazione

Durante l’esecuzione del programma ho notato che in alcune situazioni terminava o continuava anche se l’input dato era diverso da quello richiesto. Perciò ho cercato di ottimizzarlo ripetendo in ciclo alcuni comandi affinché l’input dato al programma fosse giusto.

Aggiungendo un **default** nello **switch case** eviteremo di far terminare il programma se l'input dato sarà diverso da A/a, B/b o C/c.
Inoltre il **default** ti rimanderà alla scelta dell'operazione, grazie al comando **goto**, scrivendo sull'esecuzione 'Scelta non valida'.

```
switch(scelta)
//con lo switch andrò a scegliere quale operazione andro a fare rispetto alla lettera che digiterò
    {
    case 'A':
    case 'a':
        moltiplica();
        break;
    case 'B':
    case 'b':
        dividi();
        break;
    case 'C':
    case 'c':
        ins_string(MAX_LEN);
        break;

    default:
        printf("Scelta non valida\n");
        goto start_menu;
        break;
/*nel caso la lettera sara diversa da a,b,c minusc/maiusc apparirà la scritta
'scelta non valida' e ritornerà alla scelta dell'operazione*/
}
```

```
char scelta = {'\0'};
start_menu:
menu();
scanf("%s", &scelta);
```

```c
void dividi()
{
    int a,b = 0;

    printf("Inserisci il numeratore\n");
    scanf("%d", &a);

    do{
        printf("Inserisci il denominatore\n");
        scanf("%d", &b);
        if(b == 0)
            printf("ERRORE! Prova a cambiare il denominatore con uno diverso da 0\n\n");
        }while(b == 0);

    int divisione = a/b;
    printf("La divisione tra %d e %d e' %d\n",a,b,divisione);
}
```

Con il **do while** nella divisione evito di poter dividere un numeratore per **0**. Con questo ciclo impongo al programma di dover ripetere la scelta del denominatore se dovesse essere uguale a **0**

```c
void ins_string(int maxLen) {

    char stringa[maxLen];
    char *input = (char *)malloc(maxLen * sizeof(char));
    //creo uno spazio nella memoria in cui posso mettere una stringa di caratteri con una lunghezza massima di maxLen

    while (!isValidStringLen(input, maxLen)) {
        printf("\nStringa non valida\n");
        }

    printf ("\nStringa valida");

}

int isValidStringLen(char *input, int maxLen) {
    printf("\nInserisci la stringa (massimo %d caratteri): ", maxLen);
    scanf("%s", input);

    return strlen(input) ⩽ maxLen ? 1 : 0;
    /*controllo se input è ⩽ a 10(maxLen) nel caso fosse ⩽ 10, il comnado risponderà '1',
    finisce il programma con la scritta 'stringa valida', nel caso fosse > 10 risponderà '0'
    e attiva il while che mi farà reinserire una nuova stringa*/
}
```

```c
int MAX_LEN = 10;
void menu ();
void moltiplica ();
void dividi ();
void ins_string (int maxLen);

int isValidStringLen (char * input , int maxLen);
```

Do un valore che rappresenterà il massimo numero di caratteri che posso scrivere sulla stringa. Poi come ho scritto nel commento creo uno spazio nella memoria, grazie a **malloc**, in cui posso cui mettere una stringa di caratteri con una lunghezza massima di maxLen. Faccio un controllo, se l'input dato è minore o uguale a 10 il programma termina dandoti la scritta **'Stringa valida'**, invece se si superano i 10 caratteri uscirà la scritta **'Stringa non valida'** e si ritornerà all'immissione della stringa