ANCORA UN PO' DI PRATICA

In questo Workshop riprendiamo a parlare di programmazione realizzando alcuni programmi che ci permetteranno di fare pratica con l'uso di variabili e operatori.

Nei fascicoli precedenti in cui abbiamo affrontato gli argomenti inerenti alla programmazione siamo arrivati a descrivere in maniera teorica come è possibile **dichiarare le variabili** e quali sono le funzioni offerte dai numerosi **operatori** del linguaggio C. In questo fascicolo metteremo assieme quanto visto, provando a scrivere e compilare alcuni programmi di esempio. Prima di passare ai veri e propri esempi di programmazione, però, soffermiamoci su due funzioni che ci saranno indispensabili per realizzarli: la '**printf**' e la '**scanf**'.

LA FUNZIONE PRINTF

Abbiamo già incontrato la funzione **printf** nel corso del Workshop 16, sperimentando come tale funzione ci permetta di **visualizzare sullo schermo** una stringa di testo più o meno complessa. Tuttavia, il suo funzionamento non è limitato alla stampa di stringhe, ma prevede anche la possibilità di '**formattare**' il testo di uscita inserendo **valori di variabili** e di **dati generati durante l'esecuzione del programma**.

La chiave di questo meccanismo è legata all'utilizzo di speciali '**simboli segnaposto**', che in fase di esecuzione vengono sostituiti con le variabili inserite come parametro. Per chiarire meglio il funzionamento ricorriamo a un esempio. Immaginiamo di avere a disposizione una **variabile intera** (ossia di tipo 'int') chiamata '**var**', dichiarata e inizializzata al valore **20**. Immaginiamo, ora, di voler visualizzare il valore di tale variabile, inserendolo all'interno di un messaggio. Sarà sufficiente utilizzare la funzione printf come nell'esempio:

Istruzioni 1

```
int Var = 20;          /* Dichiarazione e inizializzazione della variabile */
printf("\nIl valore della variabile var è: %d", var); /* scrittura della stringa
                                        formattata con visualizzazione del valore
                                        della variabile var */
```

Vediamo nel dettaglio cosa abbiamo scritto. L'istruzione di output inizia con la **chiamata alla funzione printf**. All'interno delle parentesi riconosciamo l'inizio della stringa per via delle virgolette e riconosciamo in essa anche il **carattere '\n'**, che indica al cursore di andare a capo. Segue, poi, il **testo** che vogliamo scrivere a video: **"il valore della variabile var è: "**. Ma cosa rappresentano i caratteri **"%d"**? Costituiscono proprio il **segnaposto** citato nelle righe precedenti. In questo caso, il segnaposto indica alla funzione printf che in quella esatta posizione **dovrà essere inserito il valore di una variabile numerica di tipo intero**

['d'], variabile che deve essere indicata successivamente alla stringa utilizzando la virgola come carattere di separazione. L'output delle istruzioni precedenti sarà, quindi:

Output 1

```
il valore della variabile var è: 20
```

I simboli segnaposto variano in base al tipo di dato che si vuole visualizzare, ma tutti sono costituiti dal carattere % seguito da una lettera (ad esempio, come visto, %d rappresenta una variabile int, %c un carattere, %s una stringa, ecc.). Per un elenco dei simboli segnaposto più utilizzati, puoi fare riferimento al box a pagina 7. Ovviamente, nella stampa formattata di una stringa è possibile utilizzare anche più di un segnaposto, purché, successivamente alla stringa, vengano passate in sequenza tutte le variabili da sostituire ai caratteri di formattazione utilizzati. Nei due box sottostanti puoi osservare un esempio di codice, seguito dal relativo output, con il quale viene stampata a video una stringa formattata in cui sono inserite le variabili 'carattere' e 'variabile_intera'.

Istruzioni 2

```
char carattere = 'x';
int variabile_intera = 40;

printf("\nLa prima variabile è: %d. La seconda è: %c.", variabile_intera, carattere);
```

Output 2

```
La prima variabile è: 40. La seconda è: x.
```

LA FUNZIONE SCANF ›››

Abbiamo visto nel paragrafo precedente come sfruttare la funzione printf per formattare un testo di uscita al fine di visualizzare il contenuto delle variabili di un programma. In questo paragrafo vedremo, invece, come è possibile utilizzare l'istruzione 'scanf' per acquisire informazioni e valori di input. Poiché scanf è una funzione molto complessa da comprendere, ci limiteremo a capire come sfruttarla per eseguire semplici operazioni di acquisizione di dati dalla tastiera. La sintassi della funzione scanf è molto simile a quella della printf: come primo parametro deve essere indicata la stringa di formattazione dei dati che devono essere letti, all'interno della quale devono

essere presenti i **segnaposto di formattazione** visti in precedenza. Successivamente a questa, devono essere elencati **gli indirizzi di memoria delle variabili in cui si vogliono memorizzare i dati letti** (vedi box del Workshop 19). Come abbiamo visto, la gestione di tali indirizzi non è affidata al programmatore, ma viene gestita in automatico dal software. Come è possibile, allora, conoscerli? È sufficiente far precedere le variabili utilizzate dal carattere **'&'**, che permette di **'ricavare' l'indirizzo di memoria di allocazione della variabile che lo segue** (tralasciamo per ora la gestione delle stringhe di caratteri, per la cui comprensione sono necessarie ulteriori conoscenze teoriche). Vediamo ora un esempio di **'lettura' di un numero intero da tastiera**:

Istruzioni 3

```
int numero=0;
scanf("%d", &numero);
```

Nella prima riga dell'esempio appena presentato abbiamo **dichiarato e inizializzato a '0'** la variabile **'numero'** di tipo **int**. La seconda riga, invece, mostra l'utilizzo della funzione **scanf**. Il primo parametro, come detto in precedenza, è la **stringa di formattazione**, all'interno della quale stiamo indicando che vogliamo **leggere un numero intero (segnaposto %d)**. Il **secondo parametro** è, invece, l'**indirizzo di memoria della variabile** in cui vogliamo memorizzare il numero intero appena letto. Si può notare come il nome della variabile sia preceduto dal **carattere &**. **ATTENZIONE: se viene omesso il carattere & durante l'uso della funzione scanf la lettura dei dati non va a buon fine ed è molto probabile il verificarsi di arresti del programma in esecuzione.** La gestione e l'acquisizione delle stringhe di caratteri con la funzione scanf sarà affrontata in futuro, in quanto ha regole leggermente differenti da quelle viste ora.

DOVE LE TROVO? ›››
Sia la funzione printf, che la funzione scanf sono messe a disposizione della **libreria stdio.h**. Ricordati di includere tale libreria iniziando il programma con l'istruzione:

Inclusione della libreria

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

Vediamo, per concludere, un semplicissimo esempio di lettura e scrittura di un numero intero:

Esempio 1

```
#include <stdio.h>

int main()
{
        int numero_da_leggere;  /* Variabile per la lettura */

        printf("\nScrivi un numero: "); /* Messaggio di richiesta inserimento */

        scanf("%d", &numero_da_leggere); /* Lettura del numero */

        printf("\nHai scritto il numero: %d", numero_da_leggere); /* Visualizzazione del
                                                                   numero letto*/

}
```

I SEGNAPOSTO PER LA FORMATTAZIONE DELLE STRINGHE ›››

Nella tabella sottostante viene mostrato un elenco semplificato dei principali simboli segnaposto utilizzabili per formattare le stringhe. Potrai vederli applicati concretamente nel corso degli esempi e dei prossimi Workshop che tratteranno di programmazione.

Segnaposto	Tipo associato
%d	Variabile numerica intera (tipo 'int').
%c	Variabile carattere (tipo 'char').
%s	Stringa di caratteri.
%f	Variabile numerica decimale (float e double). Questo particolare segnaposto può essere utilizzato secondo la forma %.xf dove x rappresenta il numero di cifre decimali che si vogliono visualizzare. Ad esempio, %.3f indica che si vuole visualizzare la variabile approssimata alla terza cifra decimale.
%u	Variabile numerica intera senza segno ('unsigned').
%%	In realtà il doppio carattere di percentuale non costituisce un segnaposto, bensì il modo con cui deve essere indicato il carattere % all'interno di una stringa quando lo si vuole stampare a video.

STEPbySTEP

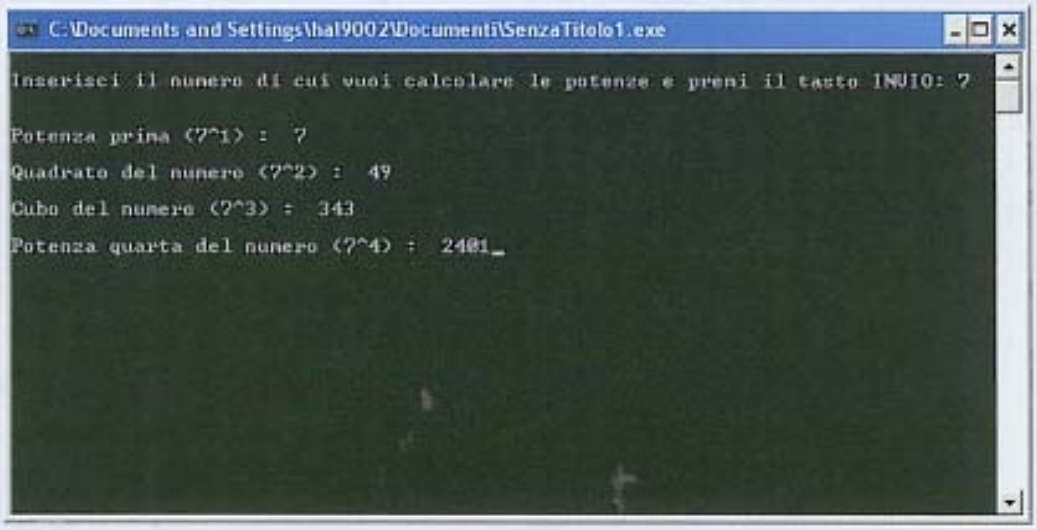
ESEMPIO 1: CALCOLARE LE POTENZE DI UN NUMERO

Questo esempio implementa un semplice programma che si occupa di leggere un numero da tastiera e stampare a video le sue potenze fino al quarto ordine.

Potenze di un numero 1

```
/* Programma di esempio numero 1 */
#include <stdio.h>
int main()
{
     int numero_letto;
     int risultato_potenza;

     /* scrive a video il messaggio di richiesta del numero */
     printf("\nInserisci il numero di cui vuoi calcolare le
      potenze e premi il tasto INVIO: ");

     /* acquisisce il numero da tastiera */
     scanf("%d", &numero_letto);

     /* calcola la potenza prima di 'numero letto'*/
     risultato_potenza = numero_letto;

     /* scrive a video il valore della potenza calcolata */
     printf("\n\nPotenza prima (%d^1) :  %d", numero_letto,
        risultato_potenza);

     /* calcola la potenza seconda di 'numero letto'*/
     risultato_potenza = numero_letto*numero_letto;

     /* scrive a video il valore della potenza calcolata */
     printf("\n\nQuadrato del numero (%d^2) :  %d", numero_letto ,
                                                  risultato_potenza);

     /* calcola la potenza terza di 'numero letto'*/
     risultato_potenza = numero_letto*numero_letto*numero_letto;

     /* scrive a video il valore della potenza calcolata */
     printf("\n\nCubo del numero (%d^3) :  %d", numero_letto,
      risultato_potenza);

     /* calcola la potenza quarta di 'numero letto'*/
     risultato_potenza = numero_letto*numero_letto*numero_letto*
        numero_letto;

     /* scrive a video il valore della potenza calcolata */
     printf("\nPotenza quarta del numero (%d^4) :  %d",
      numero_letto, risultato_potenza);

      /* utilizza le due righe successive per mantenere il
      programma in pausa finché non viene premuto il tasto INVIO */
     fflush(stdin);
     getchar(); return 1;
}
```

Nell'immagine sottostante puoi vedere la finestra di output del programma appena scritto.



ESEMPIO 2: CIRCONFERENZA E AREA DI UN CERCHIO

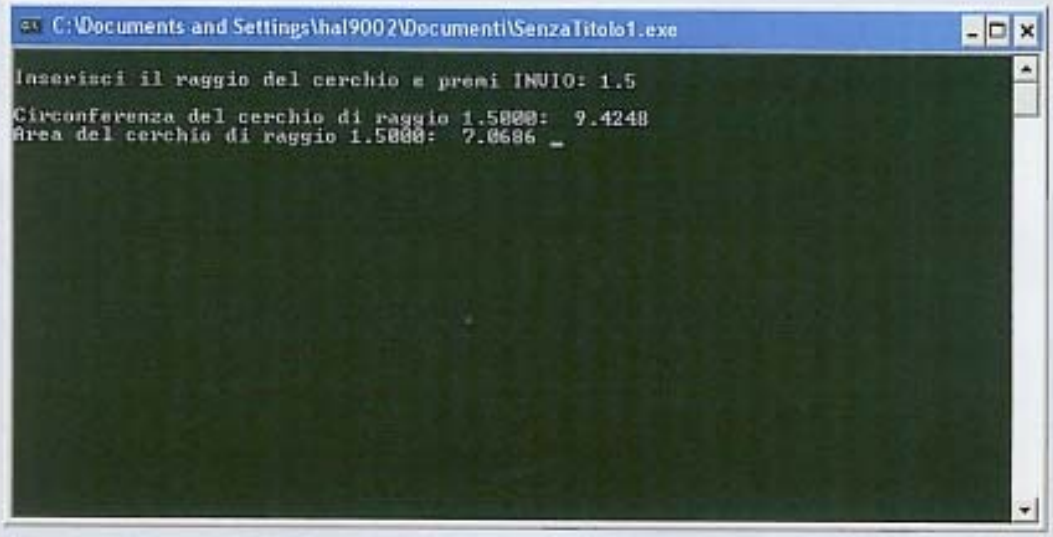
Questo secondo esempio permette di calcolare automaticamente la circonferenza e l'area di un cerchio a partire dal raggio inserito dall'utente.

Circonferenza e area di un cerchio 2

```
/* Programma di esempio numero 2 */
#include <stdio.h>
int main()
{
      /* definizione della costante PI Greco*/
      const float PI_Greco = 3.14159;

      /* definizione della variabili in cui memorizzeremo
              circonferenza e area */
      float circonferenza;
      float area;

      /* definizione della variabile raggio in cui verrà
              memorizzato il dato inserito dall'utente*/
      float raggio=0;

      /* richiesta dell'inserimento del raggio e lettura del dato
      inserito dall'utente */
      printf("\nInserisci il raggio del cerchio e premi INVIO: ");
      scanf("%f", &raggio);

      /* esegue i calcoli di circonferenza e area e memorizza i
i             risultati nelle rispettive variabili */
      circonferenza = 2 * PI_Greco * raggio;
      area = PI_Greco * raggio * raggio;
```

```
    /* visualizzazione dei risultati dei calcoli approssimando
    alla 4" cifra decimale*/
    printf("\nCirconferenza del cerchio di raggio %.4f:  %.4f ",
    raggio, circonferenza);
    printf("\nArea del cerchio di raggio %.4f:  %.4f ", raggio,
    area);

    /* utilizza le due righe successive per mantenere
    il programma in pausa finché non viene premuto il tasto
INVIO */
    fflush(stdin);
    getchar();

    return 1; /* Ritorno dalla funzione main */
}
```

Nell'immagine sottostante puoi vedere la finestra di output del programma appena scritto.



Per testare i programmi appena visti, puoi copiare i codici sorgente all'interno dell'editor di Dev-C++, procedendo poi alla compilazione e all'esecuzione in base alle procedure viste all'interno della sezione Workshop del fascicolo 17.