6 - Stringhe

Una stringa in C non è altro che una sequenza di caratteri, rappresentata in memoria come un array di tipo `char`. La caratteristica peculiare delle stringhe in C è che sono sempre terminate dal carattere speciale `\0`, che segnala la fine della stringa.

Inizializzazione

Le stringhe possono essere inizializzate in diversi modi. Si possono specificare i caratteri uno ad uno, ricordandosi però di includere manualmente il terminatore `\0`. In alternativa, si può usare una notazione più semplice, racchiudendo la stringa tra virgolette: in questo caso il terminatore viene aggiunto automaticamente dal compilatore. È una comodità che semplifica molto il lavoro.

Gestione dell'input e dell'output

Per leggere o stampare stringhe, in C si utilizzano spesso le funzioni `printf` e `scanf`, accompagnate dal placeholder `%s`. Da notare che con `scanf` non bisogna usare l'operatore di indirizzo `&`, come si fa invece con le variabili numeriche, perché una stringa è già un array. Tuttavia, è importante ricordare che `scanf` si ferma alla prima occorrenza di uno spazio, di un tab o di un invio, rendendo inadatta questa funzione per leggere stringhe contenenti spazi.

Per limitare il numero di caratteri letti in input, si può specificare un limite con una sintassi come `%9s`, che indica di leggere al massimo 9 caratteri (lasciando spazio per il terminatore). Questo accorgimento è utile per evitare che una stringa troppo lunga causi problemi di memoria.

Un esempio classico è quello di dichiarare un array di caratteri di dimensione fissata, come:

```
char string[LENGTH];
```

Poi si chiede all'utente di inserire una stringa, la si legge con `scanf`, e infine la si stampa con `puts`. Questo esempio mostra un aspetto critico del linguaggio C: se l'utente inserisce una stringa più lunga del buffer previsto, il compilatore non genera errori, ma potrebbero verificarsi comportamenti inattesi, è quindi fondamentale definire e rispettare dei limiti precisi.

Manipolazione delle stringhe*

In C, le stringhe non possono essere assegnate direttamente dopo la loro dichiarazione. Per manipolarle, si fa uso di funzioni apposite della libreria `<string.h>`. Ad esempio, si possono confrontare due stringhe con `strcmp`, copiarne una in un'altra con `strcpy`, o concatenarle

con `strcat`. Altre funzioni, come `strlen`, permettono di calcolare la lunghezza di una stringa.
Se invece si vuole analizzare o trasformare singoli caratteri, la libreria `<ctype.h>` offre strumenti utili. Ad esempio, si può verificare se un carattere è un numero (`isdigit`), una lettera (`isalpha`), o convertirlo da maiuscolo a minuscolo e viceversa con `tolower` e `toupper`.

Gestire stringhe con spazi

Un limite importante della funzione `scanf` è che non consente di leggere stringhe contenenti spazi. Per ovviare a questo problema si utilizza `fgets`, che permette di leggere stringhe con spazi inclusi specificando la dimensione massima del buffer.

Attenzione al buffer overflow

Una pratica fondamentale quando si lavora con le stringhe in C è proteggersi dal buffer overflow, che si verifica quando si scrivono più dati di quanti ne possa contenere il buffer. Questo rischio può essere mitigato limitando sempre la lunghezza dei dati in input.