Ingresso ed Uscita in C

Luca Abeni

Input e Output in C

- Linguaggio C: progettato per essere semplice e con poche istruzioni
- Non esistono neanche istruzioni di ingresso / uscita (I/O)!!!
  - Ingresso ed uscita avvengono attraverso chiamate a funzioni, non attraverso operazioni/istruzioni del linguaggio
  - Funzioni di ingresso / uscita: `printf()`, `scanf()`, ...
  - Definite in una libreria standard che è sempre linkata con tutti i programmi
  - Importante includere `stdio.h` (prototipi delle funzioni di I/O)
- Volendo, si potrebbero usare funzioni di ingresso/uscita diverse...
- ...Ma noi ci serviremo solo della libreria standard

Funzioni di I/O in C

- Servono ad implementare azioni tipo "stampa il valore di una variabilie" o "leggi un dato da tastiera"
- Sono funzioni abbastanza "strane"
  - Valore di ritorno non esistente / spesso ignorato
  - Interazione con l'ambiente esterno
  - Effetti collaterali!
- Abbiamo visto `printf()` e `scanf()`
  - Numero di parametri variabile
  - `scanf()` deve modificare i parametri
    - Apparente contraddizione con il passaggio per valore
    - Simbolo "&"

Funzione printf()

- È la funzione di output più usata in C
- Fornita dalla libreria standard del C
- Numero variabile di parametri
  - Il primo parametro è sempre una stringa
  - `printf()` visualizza tale stringa sullo schermo
  - Forma più semplice: `printf("Hi there!")`
- La stringa passata come primo parametro può contenere specificatori di formato
  - Sequenze di caratteri che cominciano con "%"
  - Sostituiti da `printf()` con un valore specificato come parametro

Specificatori di Formato in printf()

- Possono apparire nella stringa passata come primo parametro
- Per ogni specificatore di formato presente nella stringa, `printf()` accetta un parametro aggiuntivo
- Lo specificatore di formato è sostituito con il valore passato come parametro
- I caratteri che seguono "%" specificano il tipo del valore e come stamparlo
  - Esempio: `printf("v[%d]=%f", i, v[i])`
  - `i` è una variabile di tipo `int`
  - `v` è un array di `double`

Specificatori di Formato in printf()

- %d: `int`
- %u: `unsigned int`
- %x: `unsigned int` stampato come esadecimale
- %f: floating point (`float` o `double`)
- %c: `char`
- %s: stringa (array di `char` - terminato con 0)
- %ld: `long int`
- %lu: `unsigned long int`
- Nota: `ld` e `lu` $\rightarrow$ `long int` e `long unsigned int`
  - Specificatori `d` e `u` con modificatore (di lunghezza) `l`

Specificatori di Formato - 2

- In generale, uno specificatore di formato è composto da:
  - Il simbolo %
  - Un *flag* opzionale (può essere `#`, `0`, `-`, `+`, o uno spazio)
  - Una *ampiezza* opzionale: numero che specifica quanti caratteri da usare per stampare il valore
  - Una *precisione* opzionale: `.` seguito da un numero che indica quante cifre usare per la parte frazionaria dei numeri in virgola mobile
  - Un *modificatore di lunghezza* opzionale
  - Lo *specificatore di conversione* `d`, `u`, `x`, `f`, `c` o `s`

Funzione scanf()

- È una funzione di input molto usata in C
- Fornita dalla libreria standard del C
- Numero variabile di parametri (funzione variadica)
  - Il primo parametro è sempre una stringa
  - `scanf()` riceve valori dalla tastiera: la stringa specifica quali
- La stringa passata come primo parametro deve contenere almeno uno specificatore di formato
  - Sequenze di caratteri che cominciano con "%"
  - Ad ogni specificatore corrisponde un parametro aggiuntivo
  - Tale parametro specifica una variabile in cui salvare il valore letto
  - Almeno 2 parametri: stringa ed una variabile

Utilizzo di scanf()

- Caso più semplice: due parametri (stringa contenente uno specificatore di formato + variabile dove memorizzare il valore immesso)
  - Il tipo della variabile dipende dallo specificatore di formato
  - `scanf("%d", &i);` ← `i` è di tipo `int`
  - `scanf("%u", &n);` ← `n` è di tipo `unsigned int`
  - ...
- Nota: il nome della variabile è sempre preceduto da "`&`"
- Si possono usare anche più specificatori di formato
  - `scanf("%d %lf", &n, &v);` ← `n` è di tipo `int`, `v` è di tipo `double`

scanf() e Passaggio di Parametri

- `scanf()` sembra violare quanto detto riguardo al passaggio di parametri
  - In C i parametri sono passati per valore
  - Una funzione (come `scanf()`) non può modificare il valore di variabili passate come parametri attuali
  - `scanf("%u", &n);` modifica il valore di `n`... Come?
- In realtà non c'è alcuna violazione... Il "trucco" sta nel simbolo "`&`"
  - `&n` significa "indirizzo di memoria in cui sta memorizzata la variabile `n`"
  - `scanf()` non riceve come parametri le variabili, ma i loro indirizzi!!!
  - Quindi, può scrivere i valori letti in memoria, agli indirizzi specificati...

Specificatori di Formato in scanf()

- `%d`: `int`
- `%u`: `unsigned int`
- `%f`: floating point a singola precisione (`float`)
- `%c`: `char`
- `%s`: sequenza di caratteri (esclusi spazio, tab e simili), da memorizzarsi in un array di `char` terminato con 0 (stringa). L'array deve essere abbastanza grande per contenere tutti i caratteri immessi
- `%ld`: `long int`
- `%lu`: `unsigned long int`
- `%lf`: `double`

scanf() e Stringhe

- `scanf("%s", stringa);` non accetta spazi o caratteri simili...
- Come immettere stringhe contenenti anche questi caratteri?
- Varie possibilità:
  - Usare `%[...]` per specificare i caratteri che possono far parte della stringa. Il primo carattere immesso che non fa parte di quelli fra `[` e `]` termina la stringa.
    - `scanf("%[0-9]", s);` legge stringhe composte da cifre
    - `scanf("%[a-z 0-9]", s);` legge stringhe composte da caratteri minuscoli, cifre e spazi
  - Usare %c con un'ampiezza che specifica quanti caratteri leggere
    - `scanf("%5c", s);` legge esattamente 5 caratteri e li mette in un array di `char` chiamato `s`

Ancora su Specificatori di Formato

- In realtà, più complessi di come spiegato
  - Come per `printf()`, flag, ampiezza, modificatore di lunghezza e specificatore
- Alcuni specificatori sono simili a quelli usati da `printf()`
  - `%d`, `%u`, `%c`, ...
- Altri sono diversi!
  - `%f`: `float` o `double` per `printf()`, `float` per `scanf()` (`double` è `%lf`)