CORSO DI PROGRAMMAZIONE
A.A. 2014-15

Dispensa 5

Laboratorio

Dott. Mirko Ravaioli
e-mail: mirko.ravaioli@unibo.it

5 Input formattato: scanf()

La maggior parte dei programmi necessita di dati forniti dall'utente, in particolare da tastiera. Il modo più flessibile per leggere dati numerici è attraverso la funzione di libreria *scanf()*.
La funzione scanf() accetta un numero variabile di argomenti, come minimo due. Il primo è una stringa di formato che indica come interpretare l'input mediante dei caratteri speciali. Il secondo e li altri sono gli indirizzi delle variabili a cui assegnare i dati di input. Ecco un esempio:

```
scanf("%d",&numero);
```

Il primo argomento "%d", è la stringa di formato. In questo caso "%d" indica a scanf() di cercare un valore intero con segno. Il secondo argomento utilizza l'operatore di indirizzo (&) per indicare a scanf() di assegnare il valore di input alla variabile intera numero. La stringa di formato può contenere quanto segue:

- Spazi e tabulazioni che sono ignorati (possono essere utilizzati per migliorare la lettura della stringa di formato)
- Caratteri (tranne %), che sono messi in corrispondenza con caratteri diversi da spazi dell'input.
- Una o più specifiche di conversione, costituite dal carattere % seguito da caratteri speciali. In genere la stringa di formato contiene un'unica specifica di conversione per ogni variabile.

L'unica parte obbligatoria della stringa di formato è rappresentata dalle specifiche di conversione, che iniziano con % e contengono componenti opzionali e richiesti in un certo ordine. Scanf() applica specifiche in ordine ai campi di input. Un campo di input è una sequenza di caratteri diversi da spazi che termina con il successivo spazio bianco o quando l'ampiezza del campo, se specificata, viene raggiunta.

Modificatori di precisione:

Tipo	***Argomento***	***Significato***
`d`	`int *`	Intero decimale
`i`	`int *`	Intero notazione decimale, ottale (inizia con O) o esadecimale (inizia con 0x o 0X)
`o`	`int*`	Intero in notazione ottale
`u`	`unsigned int *`	Intero decimale senza segno
`c`	`char *`	Uno o più caratteri sono letti sequenzialmente nella locazione di memoria indicata dall'argomento, senza aggiungere il caarattere di terminazione \0
`s`	`char *`	Una stringa di caratteri diversi da spazi viene letta nella locazione di memoria specificata
`e,f,g`	`float *`	Un numero in virgola mobile, in notazione decimale o scientifica
`[…]`	`char *`	Una stringa, soltanto I caratteri elencati tra le parentese sono accettati. L'input termina non appena si incontra un carattere non corrispondente, oppure viene raggiunta l'ampiezza specificata o viene premuto il tasto invio, Per accettare il carattere] è necessario elencarlo per primo: […]. Un terminatore \0 viene aggiunto alla fine della stringa.

[^…]	char *	Equivale a […] salvo che sono accettati soltanto i caratteri non elencati tra parentesi.
%	nessuno	Letterale %: legge il carattere %. Non viene effettuato alcun assegnamento.

Modificatori di precisione:

h	Quando inserito prima dell'indicatore di tipo d,i,o,u,x indica che l'argomento è un puntatore di tipo short invece che un tipo int. Sui PC i due tipi coincidono perciò non serve
l	Quando inserito prima dell'indicatore di tipo d,i,o,u,x indica che l'argomento è un puntatore di tipo long. Quando inserito prima dell'indicatore e,f,g indica che l'argomento è un puntatore di tipo double
L	Quando inserito prima dell'indicatore di tipo e, f, g, indica che l'argomento è un puntatore al tipo long double

L'input di scanf() è bufferizzato, perciò non vengono ricevuti effettivamente dei caratteri finché l'utente non preme INVIO. L'intera riga di caratteri arriva poi a stdin ed è elaborata in ordine da scanf(). Il controllo torna da scanf() soltanto quando è stato ricevuto input sufficienza per soddisfare le specifiche della stringa di formato. I caratteri in più se esistono rimangono in stdin e possono causare problemi.

Quando viene eseguita una chiamata scanf() e l'utente ha inserito una singola riga, si possono avere tre situazioni: per questi esempi si assume che sia eseguita scanf("%d %d", &x, &y); per cui scanf() è in attesa di 2 decimali. Ecco le possibilità:

- La riga inserita dall'utente corrisponde alla stringa di formato. Ad esempio si supponga che l'utente immetta 12 14 e prema INVIO. In questo caso non ci sono problemi, scanf() è soddisfatta e non rimangono caratteri in stdin.
- La riga inserita dall'utente ha troppo pochi elementi per corrispondere alla stringa di formato. As esempio si supponga che l'utente immetta 12 e prema INVIO. In questo caso scanf() continua ad attendere l'input mancante e quando lo riceve l'esecuzione continua senza che rimangano dei caratteri in stdin.
- La riga inserita dall'utente ha più caratteri di quelli richiesti dalla stringa di formato. Ad esempio si supponga che l'utente immetta 12 14 16 e prema INVIO. In questo caso scanf() legge 12 e 14 e restituisce il controllo lasciando gli altri caratteri (1 e 6) in stdin.

Nella terza situazione possono verificarsi dei problemi, poiché i caratteri rimangono in attesa fino alla successiva lettura da stdin. Poi i caratteri rimasti sono i primi ad essere letti, davanti anche all'input dell'utente effettuato al momento e questo provoca errori.
Per evitare questo problema si utilizza la funzione gets() che legge caratteri rimanenti da stdin, fino al termine della riga incluso.
Per eliminare qualsiasi carattere indesiderato è possibile utilizzare la funzione fflush() passandogli come parametro stdin.

Esempio utilizzo della funzione *scanf()*:

```
#include <stdio.h>

int main()
{
      int i1, i2;
      long l1;
      double d1;

      char buf1[80], buf2[80];

      /*uso del modificatore  l per interi long e double*/

      puts("Inserire un intero e un numero in virgola mobile.");
      scanf("%ld %lf",&l1, &d1);
      printf("Hai inserito %ld e %lf.\n", l1, d1);
      /* La stringa di formato di scanf() ha utilizzato il modificatore
      l per memorizzare l'input di un tipo long e un tipo double */

      fflush(stdin);

      /*Utilizza la dimensione del campo per suddividere l'input*/

      puts("inserire un numero di 5 cifre (ad esempio 54321)");
      scanf("%2d%3d", &i1, &i2);
      /* Si noti che l'indicatore dell'ampiezza di campo nella stringa
      di formato di scanf() suddivide l'input in due valori */

      fflush(stdin);
      puts("inserire il nome e il cognome separati da uno spazio. ");
      scanf("%[^ ]%s", buf1, buf2);
      printf("\n il nome è: %s\n",buf1);
      printf("\n il cognome è: %s\n",buf2);
      /* Si noti che [^ ] nella stringa scanf() escludendo il carattere
      spazio ha causato la suddivisione dell'input */

      return 0;
}
```

La funzione fflush() viene utilizzata per eliminare qualsiasi carattere indesiderato dal flusso di input. Poiché non ci sono indicatori d'ampiezza l'intero a cinque cifre è suddiviso in due interi, uno con due caratteri ed uno con tre. L'ultimo esempio utilizza il carattere di esclusione. La stringa %[^]%s indica a scanf() di prelevare una stringa fermandosi in corrispondenza di uno spazio, in questo modo l'input viene suddiviso in maniera efficace.