

CAPITOLO 9

La formattazione dell'input/output in C

Obiettivi

- Comprendere i flussi di input e di output.
- Essere in grado di usare tutte le capacità di formattazione della stampa.
- Essere in grado di usare tutte le capacità di formattazione dell'input.
- Essere in grado di stampare utilizzando le dimensioni di campo e le precisioni.
- Essere in grado di usare i flag di formattazione della stringa di controllo del formato della `printf`.
- Essere in grado di mandare in output i letterali e le sequenze di escape.

9.1 Introduzione

Una parte importante della soluzione di ogni problema è la presentazione dei risultati. In questo capitolo discuteremo in modo approfondito le caratteristiche di formattazione di `scanf` e `printf`. Queste funzioni prendono i dati dallo *stream dello standard input* e li inviano allo *stream dello standard output*, rispettivamente. Nel Capitolo 8 sono state discusse altre quattro funzioni che utilizzano l'input e l'output standard: `gets`, `puts`, `getchar` e `putchar`. Include il file di intestazione `<stdio.h>` nei programmi che richiamano queste funzioni.

Molte delle caratteristiche di `printf` e `scanf` sono già state discusse in precedenza in questo libro. Questo capitolo riassumerà quelle caratteristiche e ne introdurrà molte altre. Il Capitolo 11 discuterà molte altre funzioni incluse nella libreria per l'input/output standard (`stdio`).

9.2 Gli stream

Tutto l'input e l'output è eseguito attraverso gli *stream*, che sono delle sequenze di byte. Nelle operazioni di input i byte fluiscono da un dispositivo (per esempio, una tastiera, un disco, una connessione di rete) verso la memoria primaria. Nelle operazioni di output i byte fluiscono dalla memoria primaria verso un dispositivo (per esempio, uno schermo, una stampante, un disco, una connessione di rete ecc.).

Nel momento in cui comincia l'esecuzione di un programma, a questo saranno connessi automaticamente tre stream. Di solito, lo stream dello standard input è connesso alla tastiera e quello dell'output allo schermo. Spesso i sistemi operativi consentono il *redirezionamento* di

questi stream verso altri dispositivi. Il terzo stream, lo *standard error*, è connesso allo schermo. I messaggi di errore saranno inviati in output sullo stream dello standard error. Gli stream saranno discussi in dettaglio nel Capitolo 11, “L’elaborazione dei file”.

9.3 Formattare l’output con printf

Con `printf` può essere ottenuta una precisa formattazione dell’output. Ogni invocazione di `printf` contiene una *stringa di controllo del formato* che descrive appunto il formato dell’output. La stringa di controllo del formato è composta da *indicatori di conversione*, *flag*, *dimensioni di campo*, *precisioni* e *caratteri letterali*. Uniti al segno di percentuale (%), questi formano le *specifiche di conversione*. La funzione `printf` ha le seguenti capacità di formattazione, ognuna delle quali sarà discussa in questo capitolo:

1. *Arrotondamento* dei valori in virgola mobile al numero indicato di cifre decimali.
2. *Allineamento* di una colonna di numeri con i separatori dei decimali che compaiono uno sull’altro.
3. *Allineamento a destra e a sinistra* degli output.
4. *Inserimento di caratteri letterali* in posizioni precise della riga inviata in output.
5. Rappresentazione dei numeri in virgola mobile in formato esponenziale.
6. Rappresentazione degli interi senza segno in formato ottale ed esadecimale. Consultate l’Appendice E, I sistemi numerici, per ottenere maggiori informazioni sui valori ottali ed esadecimale.
7. Visualizzazione di tutti i tipi di dato in campi di dimensione e precisione prefissate.

La funzione `printf` ha la forma:

```
printf (stringa di controllo del formato, altri argomenti);
```

La *stringa di controllo del formato* descrive il formato dell’output, mentre gli *altri argomenti* (che sono opzionali) corrispondono alle singole specifiche di conversione inserite nella *stringa di controllo del formato*. Ogni specifica di conversione incomincia con un segno di percentuale e termina con un indicatore di conversione. In una stringa di controllo del formato potranno essere inserite molte specifiche di conversione.



Errore tipico 9.1

Dimenticare di racchiudere tra virgolette la stringa di controllo del formato è un errore di sintassi.



Buona abitudine 9.1

Curate la formattazione dell’output in modo da ottenere una presentazione ordinata per rendere più leggibili gli output del vostro programma e ridurre gli errori dell’utente.

9.4 Visualizzare gli interi

Un intero è un numero, come 776, 0 o -52, che non contiene virgole decimali. I valori interi possono essere visualizzati in molti formati. La Figura 9.1 descrive le specifiche di conversione disponibili per gli interi.

Il programma della Figura 9.2 visualizzerà un intero utilizzando ogni indicatore di conversione disponibile. Osservate che sarà stampato soltanto il segno negativo; quelli positivi sono soppressi. Più tardi in questo capitolo vedremo in che modo sarà possibile forzare la visualizzazione dei segni positivi. Notate anche che `-455` quando sarà letto per mezzo di `%u`, verrà convertito nel valore senza segno `4294966841`.

Indicatore di conversione	Descrizione
<code>d</code>	Visualizza un intero decimale con segno.
<code>i</code>	Visualizza un intero decimale con segno. [Nota: il comportamento degli indicatori <code>i</code> e <code>d</code> sarà diverso quando saranno utilizzati con <code>scanf</code>].
<code>o</code>	Visualizza un intero ottale senza segno.
<code>u</code>	Visualizza un intero decimale senza segno.
<code>x</code> o <code>X</code>	Visualizza un intero esadecimale senza segno. <code>x</code> induce la visualizzazione delle cifre <code>0-9</code> e delle lettere <code>A-F</code> , mentre <code>X</code> induce la visualizzazione delle cifre <code>0-9</code> e delle lettere <code>a-f</code> .
<code>h</code> o <code>l</code> (lettera <code>l</code>)	Posto dinanzi a ogni indicatore di conversione per gli interi, per indicare che sarà visualizzato rispettivamente un intero <code>short</code> o <code>long</code> . Più precisamente le lettere <code>h</code> e <code>l</code> sono chiamate <i>modificatori di lunghezza</i> .

Figura 9.1 Gli indicatori di conversione per gli interi

```

1  /* Fig 9.2: fig09_02.c
2     Usare gli indicatori di conversione per gli interi */
3  #include <stdio.h>
4
5  int main()
6  {
7     printf("%d\n", 455);
8     printf("%i\n", 455); /* i è come d nella printf */
9     printf("%d\n", +455);
10    printf(„%d\n“, -455);
11    printf(„%hd\n“, 32000);
12    printf(„%ld\n“, 2000000000);
13    printf("%o\n", 455);
14    printf("%u\n", 455);
15    printf("%u\n", -455);
16    printf("%x\n", 455);
17    printf("%X\n", 455);
18
19    return 0; /* indica che il programma è terminato con successo */
20
21 } /* fine della funzione main */

```

Figura 9.2 Usare gli indicatori di conversione per gli interi (continua)

455
455
455
-455
32000
2000000000
707
455
4294966841
1c7
1C7

Figura 9.2 Usare gli indicatori di conversione per gli interi



Errore tipico 9.2

Visualizzare un valore negativo con un indicatore di conversione che si aspetta un valore unsigned.

9.5 Visualizzare i numeri in virgola mobile

Un valore in virgola mobile contiene una virgola decimale come 33,5, 0,0 o 657,983. I valori in virgola mobile potranno essere visualizzati in molti formati. La Figura 9.3 descrive gli indicatori di conversione disponibili per i numeri in virgola mobile.

Indicatore di conversione	Descrizione
e o E	Visualizza un valore in virgola mobile nella notazione esponenziale.
f	Visualizza un valore in virgola mobile nella notazione in virgola fissa.
g o G	Visualizza un valore in virgola mobile sia nel formato di f che in quello di e
	(o di E) in base alla grandezza del valore.
L	Posto dinanzi a un qualsiasi indicatore di conversione per i numeri in virgola mobile, per indicare che sarà visualizzato un valore in virgola mobile long double.

Figura 9.3 Gli indicatori di conversione per i valori in virgola mobile

Gli indicatori di conversione e ed E visualizzano i valori in virgola mobile nella *notazione esponenziale*. Nei computer, la notazione esponenziale è l'equivalente della *notazione scientifica* utilizzata in matematica. Per esempio, il valore 150,4582 nella notazione scientifica è rappresentato come

$$1,504582 \times 10^2$$

mentre in quella esponenziale è rappresentato dal computer come

1,504582E+02

Questa notazione indica che 1,504582 è moltiplicato per 10 elevato alla seconda potenza (E+02). La E sta appunto per “esponente”.

I valori visualizzati con gli indicatori di conversione e, E e f saranno visualizzati per default con 6 cifre di precisione a destra della virgola decimale (per esempio, 1,04592); le altre precisioni potranno essere specificate in modo esplicito. L'indicatore di conversione f visualizza sempre almeno una cifra a sinistra della virgola decimale. Gli indicatori di conversione e ed E visualizzano rispettivamente una e minuscola e una E maiuscola davanti all'esponente, e stamperanno sempre e solo una cifra a sinistra della virgola decimale.

L'indicatore di conversione g (G) visualizza i valori nei formati e (E) o f, ma eliminando gli zeri finali nella parte decimale (in altri termini, 1,234000 sarebbe visualizzato come 1,234). I valori saranno visualizzati nel formato e (E) qualora, dopo la loro conversione in notazione esponenziale, l'esponente sia inferiore a -4 oppure sia maggiore o uguale alla precisione specificata (per default 6 cifre significative per g e G). In caso contrario, per visualizzare i valori sarà utilizzato l'indicatore di conversione f. Nella parte frazionaria di un valore inviato in output con g o G gli zeri finali non saranno visualizzati. Perché sia visualizzata la virgola dei decimali sarà necessaria almeno una cifra decimale. Con l'indicatore di conversione g, i valori 0,0000875, 8750000,0, 8,75, 87,50 e 875 sarebbero visualizzati come 8,75e-05, 8,75e+06, 8,75, 87,5 e 875. Il valore 0,0000875 utilizzerà la notazione e perché, quando sarà stato convertito in quella esponenziale, il suo esponente (-5) sarà minore di -4. Il valore 8750000,0 utilizzerà la notazione e perché il suo esponente (6) sarà uguale alla precisione di default.

Per gli indicatori di conversione g e G, la precisione indica il numero massimo di cifre significative che saranno visualizzate, inclusa quella a sinistra della virgola decimale. Utilizzando la specifica di conversione %g, il valore 1234567,0 sarà visualizzato come 1,23457e+06 (ricordate che tutti gli indicatori di conversione per i numeri in virgola mobile hanno per default una precisione di 6 cifre). Osservate che nel risultato ci saranno appunto 6 cifre significative. Nel caso che il valore sia visualizzato in notazione esponenziale, la differenza tra g e G sarà la stessa che intercorre tra e ed E: la g minuscola visualizzerà una e minuscola, mentre la G maiuscola stamperà una E maiuscola.

Il programma della Figura 9.4 mostrerà l'utilizzo delle tre specifiche di conversione disponibili per i valori in virgola mobile. Osservate che le specifiche di conversione %E, %e e %g arrotonderanno il valore inviato in output, mentre la specifica di conversione %f non lo farà.

```

1  /* Fig 9.4: fig09_04.c */
2  /* Visualizzare i numeri in virgola mobile con gli indicatori
3     di conversione per i valori in virgola mobile */
4
5  #include <stdio.h>
6
7  int main()
8  {
9     printf("%e\n", 1234567.89);
10    printf("%e\n", +1234567.89);

```

Figura 9.4 Usare gli indicatori di conversione per i valori in virgola mobile (continua)

```

11     printf("%e\n", -1234567.89);
12     printf("%E\n", 1234567.89);
13     printf("%f\n", 1234567.89);
14     printf("%g\n", 1234567.89);
15     printf("%G\n", 1234567.89);
16
17     return 0; /* indica che il programma è terminato con successo */
18
19 } /* fine della funzione main */

```

```

1.234568e+06
1.234568e+06
-1.234568e+06
1.234568E+06
1234567.890000
1.23457e+06
1.23457E+06

```

Figura 9.4 Usare gli indicatori di conversione per i valori in virgola mobile



Collaudo e messa a punto 9.1

Quando inviate i dati in output, assicuratevi di informare l'utente delle situazioni in cui i dati potrebbero non essere precisi a causa della formattazione (per esempio, per gli errori di arrotondamento dovuti alla precisione specificata).

9.6 Visualizzare le stringhe e i caratteri

Gli indicatori di conversione `c` e `s` sono utilizzati rispettivamente per visualizzare i singoli caratteri e le stringhe. L'indicatore di conversione `c` richiede un argomento di tipo `char`, mentre `s` richiede un puntatore a `char`. L'indicatore di conversione `s` provocherà la visualizzazione dei caratteri finché non sarà stato incontrato il carattere nullo (`'\0'`) di terminazione della stringa. Il programma mostrato nella Figura 9.5 visualizzerà alcuni caratteri e stringhe utilizzando gli indicatori di conversione `c` e `s`.

```

1  /* Fig 9.5: fig09_05.c */
2  /* Visualizzare stringhe e caratteri */
3  #include <stdio.h>
4
5  int main()
6  {
7      char character = 'A'; /* inizializza la variabile di tipo char */
8      char string[] = "This is a string"; /* inizializza
9                                     il vettore di elementi di tipo char */
10     const char *stringPtr = "This is also a string";
11                                     /* puntatore a char */
12
13     printf("%c\n", character);

```

Figura 9.5 Utilizzare gli indicatori di conversione per i caratteri e le stringhe (continua)

```

12     printf("%s\n", "This is a string");
13     printf("%s\n", string);
14     printf("%s\n", stringPtr);
15
16     return 0; /* indica che il programma è terminato con successo */
17
18 } /* fine della funzione main */
    
```

```

A
This is a string
This is a string
This is also a string
    
```

Figura 9.5 Utilizzare gli indicatori di conversione per i caratteri e le stringhe



Errore tipico 9.3

Utilizzare %c per visualizzare una stringa è un errore. La specifica di conversione %c si attende un argomento di tipo char. Una stringa invece è un puntatore a char (ossia un char *).



Errore tipico 9.4

Utilizzare %s per visualizzare un argomento di tipo char, su alcuni sistemi, provocherà un errore fatale detto violazione di accesso. La specifica di conversione %s si aspetta un argomento di tipo puntatore a char.



Errore tipico 9.5

È un errore di sintassi usare degli apici singoli per delimitare le stringhe di caratteri. Queste, infatti, devono essere racchiuse da virgolette.



Errore tipico 9.6

Utilizzare le virgolette intorno alle costanti di carattere creerà in realtà una stringa formata da due caratteri, il secondo dei quali sarà il carattere nullo di terminazione. Una costante di carattere è invece un singolo carattere racchiuso tra apici singoli.

9.7 Gli altri indicatori di conversione

Gli ultimi tre indicatori di conversione rimasti sono: p, n e % (Figura 9.6).

Indicatore di conversione	Descrizione
p	Visualizza il valore di un puntatore in un formato definito dall'implementazione.
n	Immagazzina il numero di caratteri già inviato in output dall'istruzione printf corrente. Come argomento corrispondente dovrà essere fornito un puntatore a un intero. L'indicatore di conversione non visualizzerà niente.
%	Visualizza il simbolo di percentuale.

Figura 9.6 Gli altri indicatori di conversione



Obiettivo portabilità 9.1

L'indicatore di conversione `p` visualizza l'indirizzo di un puntatore in un formato definito dall'implementazione (su molti sistemi, è utilizzata la notazione esadecimale piuttosto che quella decimale).

L'indicatore di conversione `n` immagazzina il numero di caratteri già inviati in output dall'istruzione `printf` corrente: infatti, l'argomento corrispondente è un puntatore alla variabile intera nella quale sarà immagazzinato il valore. La specifica di conversione `%n` non visualizza niente. L'indicatore di conversione `%` visualizza un segno di percentuale.

In Figura 9.7, `%p` visualizzerà il valore di `ptr` e l'indirizzo di `x`; questi dati saranno identici poiché l'indirizzo di `x` sarà stato assegnato a `ptr`. In seguito, `%n` immagazzinerà nella variabile intera `y` il numero dei caratteri che saranno stati inviati in output dalla terza istruzione `printf` (riga 15) e il valore di `y` sarà visualizzato. L'ultima istruzione `printf` (riga 21) utilizzerà `%%` per visualizzare il carattere `%` all'interno di una stringa. Osservate che ogni chiamata di `printf` restituisce un valore: il numero dei caratteri inviati in output o un valore negativo, qualora si sia verificato un errore nell'output.



Errore tipico 9.7

Tentare di visualizzare il simbolo di percentuale usando `%` invece di `%%` nella stringa di controllo del formato. Nel momento in cui in una stringa di controllo del formato appare un `%`, questo deve essere seguito da un indicatore di conversione.

```

1  /* Fig 9.7: fig09_07.c */
2  /* Usare gli indicatori di conversione p, n, e % */
3  #include <stdio.h>
4
5  int main()
6  {
7      int *ptr;          /* dichiara un puntatore al tipo int */
8      int x = 12345;     /* inizializza la variabile x di tipo int
9      int y;             /* dichiara la variabile y di tipo int */
10
11     ptr = &x;          /* assegna l'indirizzo di x a ptr */
12     printf("The value of ptr is %p\n", ptr);
13     printf("The address of x is %p\n\n", &x);
14
15     printf("Total characters printed on this line is:%n", &y);
16     printf(" %d\n\n", y);
17
18     y = printf("This line has 28 characters\n");
19     printf("%d characters were printed\n\n", y);
20
21     printf("Printing a %% in a format control string\n");
22
23     return 0; /* indica che il programma è terminato con successo */
24
25 } /* fine della funzione main */

```

Figura 9.7 Utilizzare gli indicatori di conversione `p`, `n` e `%` (continua)

```

The value of ptr is 0012FF78
The address of x is 0012FF78

Total characters printed on this line is: 38

This line has 28 characters
28 characters were printed

Printing a % in a format control string

```

Figura 9.7 Utilizzare gli indicatori di conversione p, n e %

9.8 Visualizzare con le dimensioni di campo e le precisioni

L'esatta misura del campo in cui saranno visualizzati i dati è specificata dalla cosiddetta *dimensione di campo*. Nel caso che la dimensione del campo sia maggiore del dato da visualizzare questo, di norma, sarà allineato a destra all'interno di quel campo. Normalmente l'intero che rappresenta la dimensione del campo è inserito tra il segno di percentuale (%) e l'indicatore di conversione (ad esempio, %4d). Il programma della Figura 9.8 visualizzerà due gruppi di cinque numeri, allineando a destra quelli che contengono meno cifre della dimensione del campo. Osservate che, per visualizzare dei valori con una dimensione maggiore di quella del campo, questa sarà incrementata, e che il segno dei valori negativi utilizzerà una posizione all'interno della stessa. Le dimensioni dei campi potranno essere utilizzate con tutti gli indicatori di conversione.



Errore tipico 9.8

Non fornire una dimensione di campo sufficientemente grande per gestire il valore da visualizzare potrebbe spostare gli altri dati da visualizzare e produrre degli output disordinati. Studiate i vostri dati!

```

1  /* Fig 9.8: fig09_08.c */
2  /* Visualizzare gli interi allineati a destra */
3  #include <stdio.h>
4
5  int main()
6  {
7      printf("%4d\n", 1);
8      printf("%4d\n", 12);
9      printf("%4d\n", 123);
10     printf("%4d\n", 1234);
11     printf("%4d\n\n", 12345);
12
13     printf("%4d\n", -1);
14     printf("%4d\n", -12);
15     printf("%4d\n", -123);

```

Figura 9.8 Allineare a destra in un campo gli interi (continua)

```

16     printf("%4d\n", -1234);
17     printf("%4d\n", -12345);
18
19     return 0; /* indica che il programma è terminato con successo */
20
21 } /* fine della funzione main */

```

```

 1
 12
123
1234
12345

-1
-12
-123
-1234
-12345

```

Figura 9.8 Allineare a destra in un campo gli interi

La funzione `printf` fornisce anche la possibilità di specificare la *precisione* con cui il dato dovrà essere visualizzato. La precisione ha un significato diverso per i vari tipi di dato. Utilizzata insieme agli indicatori di conversione per gli interi, la precisione indica il numero minimo di cifre da visualizzare. Nel caso che il valore visualizzato contenga meno cifre di quelle indicate dalla precisione, davanti al suddetto valore saranno inseriti degli zeri finché il numero totale delle cifre non risulti equivalente a quello della precisione. La precisione predefinita per gli interi è 1. Utilizzata con gli indicatori di conversione `e`, `E` e `f` per i valori in virgola mobile, la precisione indica il numero di cifre che dovranno comparire dopo la virgola dei decimali. Utilizzata con gli indicatori di conversione `g` e `G`, la precisione indica il numero massimo di cifre significative da visualizzare. Utilizzata con l'indicatore di conversione `s`, la precisione indica il numero massimo di caratteri della stringa che dovranno essere scritti. Per utilizzare la precisione, inserirete un punto (`.`) seguito dall'intero che rappresenta la precisione, tra il segno di percentuale e l'indicatore di conversione. In Figura 9.9 verrà mostrato l'utilizzo della precisione nelle stringhe di controllo del formato. Osservate che un valore in virgola mobile sarà arrotondato, qualora sia visualizzato con una precisione inferiore al numero originale delle sue cifre decimali.

```

1  /* Fig 9.9: fig09_09.c */
2  /* Usare la precisione mentre si visualizzano gli interi,
3     i numeri in virgola mobile e le stringhe */
4  #include <stdio.h>
5
6  int main()
7  {
8     int i = 873;          /* inizializza la variabile I di tipo int */

```

Figura 9.9 Utilizzare le precisioni per visualizzare informazioni di tipo differente (continua)

```

9      float f = 123.94536; /* inizializza la variabile f
                                di tipo double */
10     char s[] = "Happy Birthday"; /* inizializza il vettore
                                di caratteri s */
11
12     printf("Using precision for integers\n");
13     printf("\t%.4d\n\t%.9d\n\n", i, i);
14
15     printf("Using precision for floating-point numbers\n");
16     printf("\t%.3f\n\t%.3e\n\t%.3g\n\n", f, f, f);
17
18     printf("Using precision for strings\n");
19     printf("\t%.11s\n", s);
20
21     return 0; /* indica che il programma è terminato con successo */
22
23 } /* fine della funzione main */

```

```

Using precision for integers
    0873
    000000873

Using precision for floating-point numbers
    123.945
    1.239e+002
    124

Using precision for strings
    Happy Birth

```

Figura 9.9 Utilizzare le precisioni per visualizzare informazioni di tipo differente

La dimensione del campo e la precisione potranno essere combinate inserendo, tra il segno di percentuale e l'indicatore di conversione, la dimensione del campo seguita dal punto e dalla precisione, come nell'istruzione

```
printf("%9.3f", 123.456789);
```

che visualizzerà il valore 123,457 con tre cifre dopo la virgola dei decimali e allineato a destra in un campo di nove posizioni.

Sarà anche possibile specificare la dimensione di campo e la precisione, utilizzando delle espressioni intere inserite nella lista degli argomenti che segue la stringa di controllo del formato. Per utilizzare questa caratteristica, inserirete un * (asterisco) al posto della dimensione di campo o della precisione (o di entrambe). In sostituzione dell'asterisco sarà valutato e utilizzato l'argomento di tipo `int` corrispondente. Un valore per la dimensione del campo potrà essere positivo o negativo (in quest'ultimo caso ciò provocherà l'allineamento a sinistra dell'output, come descritto nella sezione successiva). L'istruzione

```
printf("%*.2f", 7, 2, 98.736);
```

utilizzerà 7 per la dimensione del campo, 2 per la precisione e visualizzerà il valore 98,74 allineato a destra.