Capitolo 9 - Input/Output formattato

Obiettivi

- In questo capitolo, impareremo a:
  - Capire gli stream di input e di output.
  - Utilizzare la formattazione di stampa.
  - Utilizzare l'input formattato.

Introduzione

- In questo capitolo
  - Presentazione dei risultati
  - `scanf` e `printf`
  - Gli Stream (input e output)
    - `gets`, `puts`, `getchar`, `putchar` (in `<stdio.h>`)

Gli Stream

- Gli Stream
  - Sequenze di caratteri organizzate in linee
    - Ogni linea consiste di zero o più caratteri che termina con un carattere newline
    - L'ANSI C supporta linee di almeno 254 caratteri
  - Effettuano input e output
  - Possono essere riderizionati
    - Standard input – keyboard
    - Standard output – screen
    - Standard error – screen

Formattare l'Output con la `printf`

- `printf`
  - Output formattato
    - Specifiche di conversione: flags, dimensioni di campo, precisione, ecc.
  - Può effettuare l'arrotondamento, l'allineamento delle colonne, giustificazione destra/sinistra, inserimento di caratteri letterale, formato esponenziale, formato esadecimale, e larghezza fissa e precisione
- Formattazione
  - `printf(` *format-control-string, other-arguments* `);`
  - Format control string: descrive il formato di output
  - Other-arguments: corrispondono ad ogni specifica di conversione nella format-control-string
    - Ogni specifica inizia con il simbolo di percentuale (%), e termina con l'indicatore di conversione

© Copyright 1992–2004 by Deitel & Associates, Inc. and Pearson Education Inc. All Rights Reserved.

Stampa di interi

Conversion Specifier	Description
d	Display a signed decimal integer.
i	Display a signed decimal integer. (*Note:* The i and d specifiers are different when used with scanf.)
o	Display an unsigned octal integer.
u	Display an unsigned decimal integer.
x or X	Display an unsigned hexadecimal integer. X causes the digits 0-9 and the letters A-F to be displayed and x causes the digits 0-9 and a-f to be displayed.
h or l (letter l)	Place before any integer conversion specifier to indicate that a short or long integer is displayed respectively. Letters h and l are more precisely called *length modifiers*.

Fig. 9.1 Integer conversion specifiers.

© Copyright 1992–2004 by Deitel & Associates, Inc. and Pearson Education Inc. All Rights Reserved.

Stampa di interi

- Gli interi
  - Intero numero (nessun punto decimale): 25, 0, -9
  - Positivo, negativo, o zero
  - Solo il segno meno viene stampato di default

© Copyright 1992–2004 by Deitel & Associates, Inc. and Pearson Education Inc. All Rights Reserved.

Outline

fig09_02.c

```c
1  /* Fig 9.2: fig09_02.c */
2  /* Using the integer conversion specifiers */
3  #include <stdio.h>
4
5  int main()
6  {
7     printf( "%d\n", 455 );
8     printf( "%i\n", 455 );  /* i same as d in 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; /* indicates successful termination */
20
21  } /* end main */
```

© Copyright 1992–2004 by Deitel & Associates, Inc. and Pearson Education Inc. All Rights Reserved.

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

Program Ouptut

Stampare i numeri Floating-Point

- Numeri Floating Point
  - Hanno un punto decimale (33.5)
  - Notazione esponenziale (la versione scientifica dei computer)
    - 150.3 è 1.503 x 10^2 in versione scientifica
    - 150.3 è 1.503E+02 in versione esponenziale (E sta per esponente)
    - usa e o E
  - f – stampa floating point con almeno una cifra a sinistra del decimale
  - g (o G) – stampa in f o e senza zero superflui (1.2300 diventa 1.23)
    - Usa l'esponenziale se l'esponente è minore di -4, o maggiore o uguale alla precisione (6 cifre di default)

Stampa di numeri Floating-Point

Conversion specifier	Description
e or E	Display a floating-point value in exponential notation.
f	Display floating-point values.
g or G	Display a floating-point value in either the floating-point form f or the exponential form e (or E).
L	Place before any floating-point conversion specifier to indicate that a longdouble floating-point value is displayed.

Fig. 9.3   Floating-point conversion specifiers.

```c
1   /* Fig 9.4: fig09_04.c */
2   /* Printing floating-point numbers with
3      floating-point conversion specifiers */
4
5   #include <stdio.h>
6
7   int main()
8   {
9      printf( "%e\n", 1234567.89 );
10     printf( "%e\n", +1234567.89 );
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; /* indicates successful termination */
18
19  } /* end main */
```

fig09_04.c

Program Output

```
1.234568e+006
1.234568e+006
-1.234568e+006
1.234568E+006
1234567.890000
1.23457e+006
1.23457E+006
```

9.6 Stampa di stringhe e caratteri

- c
  - Stampa argomenti char
  - Non può essere utilizzato per stampare il primo carattere di una stringa
- s
  - Richiede un puntatore a char come argomento
  - Stampa caratteri fino a NULL ('\0')
  - Non può stampare un argomento char
- Ricordare che
  - I singoli apici per caratteri costante ('z')
  - Doppi apici per le stringhe "z" (che in reltà contiene due caratteri, 'z' e '\0')

```
1  /* Fig 9.5: fig09_05c */
2  /* Printing strings and characters */
3  #include <stdio.h>
4
5  int main()
6  {
7     char character = 'A'; /* initialize char */
8     char string[] = "This is a string"; /* initialize char array */
9     const char *stringPtr = "This is also a string"; /* char pointer */
10
11     printf( "%c\n", character );
12     printf( "%s\n", "This is a string" );
13     printf( "%s\n", string );
14     printf( "%s\n", stringPtr );
15
16     return 0; /* indicates successful termination */
17
18  } /* end main */
```

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

Altri indicatori di conversione

- p
  - Visualizza il valore di un puntatore (indirizzo)
- n
  - Memorizza il numero di caratteri già visualizzati dalla printf
  - Ha un puntatore ad un intero come argomento
  - La specifica %n non stampa niente
  - Ogni chiamata a printf restituisce un valore
    - Numero di caratteri
    - Numero negativo in caso di errore
- %
  - Stampa il simbolo di percentuale
  - %%

Altri indicatori di conversione

Conversion specifier	Description
p	Display a pointer value in an implementation-defined manner.
n	Store the number of characters already output in the current printf statement. A pointer to an integer is supplied as the corresponding argument. Nothing is displayed.
%	Display the percent character.

Fig. 9.6   Other conversion specifiers.

```
1  /* Fig 9.7: fig09_07.c */
2  /* Using the p, n, and % conversion specifiers */
3  #include <stdio.h>
4
5  int main()
6  {
7     int *ptr;      /* define pointer to int */
8     int x = 12345; /* initialize int x */
9     int y;         /* define int y */
10
11    ptr = &x;      /* assign address of x to 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:%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; /* indicates successful termination */
24
25  } /* end main */
```

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

Total characters printed on this line: 38

This line has 28 characters
28 characters were printed

Printing a % in a format control string
```

Precisione

- Dimensioni di campo
  - La dimensione di campo da stampare
  - Se la dimensione è maggiore del dato viene utilizzato il valore di default
    - Se la dimensione del campo è troppo piccola, viene aumentata per contenere il dato
    - Il segno meno usa una posizione carattere nel campo
  - Le dimensioni intere inserite fra % e l'indicatore di conversione
  - %4d – dimensione di campo 4

Precisione

- Precisione
  - Il significato varia in base al tipo di dato
  - Interi (default 1)
    - Minumo numero di cifre da stampare
      - Se il dato è troppo piccolo, viene prefisso da zeri
  - Floating point
    - Numoro di cifre che devono apparire dopo il decimale (e e f)
      - per g – massimo numero di cifre significative
  - Stringhe
    - Massimo numero di caratteri della stringa da stampare
  - Formato
    - Utilizzare un punto (.) prima della precisione
      %. 3f

Precisione

- Dimensione di campo e precisione
  - Possono essere specificati entrambi
    - %width.precision
      %5.3f
  - dimensione di campo negativa – giustificato a sinistra
  - dimensione di campo positiva – giustificato a destra
  - La precisione deve essere positiva
  - Possono essere utilizzati gli interi per determinare la dimensione di campo e la preciosione
    - Posizionare l'asterisco (*) in luogo della dimensione di campo o della precisione
      - Confrontato ad un argomento int nella lista degli argomenti
    - Esempio:
      printf("%*.*f", 7, 2, 98.736);

Outline

fig09_08.c

```c
1   /* Fig 9.8: fig09_08.c */
2   /* Printing integers right-justified */
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 );
16     printf( "%4d\n", -1234 );
17     printf( "%4d\n", -12345 );
18
19     return 0; /* indicates successful termination */
20
21  } /* end main */
```

Outline

Program Output

```
   1
  12
 123
1234
12345

  -1
 -12
-123
-1234
-12345
```

Outline

fig09_09.c

```c
1   /* Fig 9.9: fig09_09.c */
2   /* Using precision while printing integers,
3      floating-point numbers, and strings */
4   #include <stdio.h>
5
6   int main()
7   {
8      int i = 873;              /* initialize int i */
9      double f = 123.94536;     /* initialize double f */
10     char s[] = "Happy Birthday"; /* initialize char array 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; /* indicates successful termination */
22
23  } /* end 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
```

△
▽ Outline

Program Output

Uso di Flag in `printf`

- Flag
  - Ulteriori possibilità di formattazione
  - Posizionare un flag immediatamente a destra del simbolo percentuale
  - È possibile combinare più flag

Flag	Description
– (minus sign)	Left-justify the output within the specified field.
+ (plus sign)	Display a plus sign preceding positive values and a minus sign preceding negative values.
space	Print a space before a positive value not printed with the + flag.
#	Prefix 0 to the output value when used with the octal conversion specifier o.
	Prefix 0x or 0X to the output value when used with the hexadecimal conversion specifiers x or X.
	Force a decimal point for a floating-point number printed with e, E, f, g or G that does not contain a fractional part. (Normally the decimal point is only printed if a digit follows it.) For g and G specifiers, trailing zeros are not eliminated.
0 (zero)	Pad a field with leading zeros.

Fig. 9.10 Format control string flags.

```
1   /* Fig 9.11: fig09_11.c */
2   /* Right justifying and left justifying values */
3   #include <stdio.h>
4
5   int main()
6   {
7      printf( "%10s%10d%10c%10f\n\n", "hello", 7, 'a', 1.23 );
8      printf( "%-10s%-10d%-10c%-10f\n", "hello", 7, 'a', 1.23 );
9
10      return 0; /* indicates successful termination */
11
12  } /* end main */
```

△
▽ Outline

fig09_11.c

Program Output

```
hello          7          a  1.230000

hello      7          a          1.230000
```

```
1   /* Fig 9.12: fig09_12.c */
2   /* Printing numbers with and without the + flag */
3   #include <stdio.h>
4
5   int main()
6   {
7      printf( "%d\n%d\n", 786, -786 );
8      printf( "%+d\n%+d\n", 786, -786 );
9
10      return 0; /* indicates successful termination */
11
12  } /* end main */
```

△
▽ Outline

fig09_12.c

Program Output

```
786
-786
+786
-786
```

```
1  /* Fig 9.13: fig09_13.c */
2  /* Printing a space before signed values
3     not preceded by + or - */
4  #include <stdio.h>
5
6  int main()
7  {
8     printf( "% d\n% d\n", 547, -547 );
9
10    return 0; /* indicates successful termination */
11
12 } /* end main */
```

```
 547
-547
```

```
1  /* Fig 9.14: fig09_14.c */
2  /* Using the # flag with conversion specifiers
3     o, x, X and any floating-point specifier */
4  #include <stdio.h>
5
6  int main()
7  {
8     int c = 1427;       /* initialize c */
9     double p = 1427.0; /* initialize p */
10
11    printf( "%#o\n", c );
12    printf( "%#x\n", c );
13    printf( "%#X\n", c );
14    printf( "\n%g\n", p );
15    printf( "%#g\n", p );
16
17    return 0; /* indicates successful termination */
18
19 } /* end main */
```

```
02623
0x593
0X593

1427
1427.00
```

```
1  /* Fig 9.15: fig09_15.c */
2  /* Printing with the 0( zero ) flag fills in leading zeros */
3  #include <stdio.h>
4
5  int main()
6  {
7     printf( "%+09d\n", 452 );
8     printf( "%09d\n", 452 );
9
10    return 0; /* indicates successful termination */
11
12 } /* end main */
```

```
+00000452
000000452
```

Stampa di letterali e sequanze Escape

- Stampa di letterali
  - Possono essere stampati più caratteri
  - Problemi con alcuni caratteri come il simbolo "
  - Devono essere rappresentati con sequenze escape
    - Rappresentato da un backslash \ seguito da un carattere escape

Stampa di letterali e sequanze Escape

Escape sequence	Description
\'	Output the single quote (') character.
\"	Output the double quote (") character.
\?	Output the question mark (?) character.
\\	Output the backslash (\) character.
\a	Cause an audible (bell) or visual alert.
\b	Move the cursor back one position on the current line.
\f	Move the cursor to the start of the next logical page.
\n	Move the cursor to the beginning of the next line.
\r	Move the cursor to the beginning of the current line.
\t	Move the cursor to the next horizontal tab position.
\v	Move the cursor to the next vertical tab position.

Fig. 9.16   Escape sequences.

Formattazione dell'Input con scanf

Conversion specifier	Description
Integers	
d	Read an optionally signed decimal integer. The corresponding argument is a pointer to integer.
i	Read an optionally signed decimal, octal, or hexadecimal integer. The corresponding argument is a pointer to integer.
o	Read an \octal integer. The corresponding argument is a pointer to unsigned integer.
u	Read an unsigned decimal integer. The corresponding argument is a pointer to unsigned integer.
x or X	Read a hexadecimal integer. The corresponding argument is a pointer to unsigned integer.
h or l	Place before any of the integer conversion specifiers to indicate that a short or long integer is to be input.

Fig. 9.17   Conversion specifiers for scanf.

Formattazione dell'Input con scanf

Conversion specifier	Description
Floating-point numbers	
e, E, f, g or G	Read a floating-point value. The corresponding argument is a pointer to a floating-point variable.
l or L	Place before any of the floating-point conversion specifiers to indicate that a double or long double value is to be input.
Characters and strings	
C	Read a character. The corresponding argument is a pointer to char, no null ('\0') is added.
S	Read a string. The corresponding argument is a pointer to an array of type char that is large enough to hold the string and a terminating null ('\0') character—which is automatically added.
Scan set	
[scan characters	Scan a string for a set of characters that are stored in an array.
Miscellaneous	
P	Read an address of the same form produced when an address is output with %p in a printf statement.
N	Store the number of characters input so far in this scanf. The corresponding argument is a pointer to integer
%	Skip a percent sign (%) in the input.

Fig. 9.17 Conversion specifiers for scanf.

Formattazione dell'Input con scanf

- scanf
  - Input formattato
  - Possibilità
    - Input di tutti i tipi di dati
    - Input di specifici caratteri
    - Salta specifici caratteri
- Formattazione
  - scanf(*format-control-string*, *other-arguments*);
  - Format-control-string
    - Descrive il formato di input
  - Other-arguments
    - Puntatore alle variabili dove memorizzare l' input
  - Può includere dimensioni di campo per leggere uno specifico numero di caratteri dallo stream

Formattazione dell'Input con scanf

- Scan sets (insieme di scansione)
  - Insieme di caratteri racchiusi tra parentesi quadre [ ]
    - Preceduti dal simbolo %
  - Scandisce stream di input, facendo riferimento solo ai caratteri nello scan set
    - Se occorre un matching, memorizza il carattere nell'array specificato
    - Si ferma solo quando incontra un carattere che non è nello scan set
  - Scan sets invertiti
    - Utilizzare ^: [^aei ou]
    - Vengono memorizzati i caratteri che non si trovano nello scan set
- Skipping characters
  - Include i caratteri da saltare nel controllo del formato
  - O, usa *
    - Salta ogni tipo di carattere senza memorizzarlo

```c
1  /* Fig 9.18: fig09_18.c */
2  /* Reading Integers */
3  #include <stdio.h>
4
5  int main()
6  {
7     int a; /* define a */
8     int b; /* define b */
9     int c; /* define c */
10    int d; /* define d */
11    int e; /* define e */
12    int f; /* define f */
13    int g; /* define g */
14
15    printf( "Enter seven integers: " );
16    scanf( "%d%i%i%i%o%u%x", &a, &b, &c, &d, &e, &f, &g );
17
18    printf( "The input displayed as decimal integers is:\n" );
19    printf( "%d %d %d %d %d %d %d\n", a, b, c, d, e, f, g );
20
21    return 0; /* indicates successful termination */
22
23 } /* end main */
```

Program Output

```
Enter seven integers: -70 -70 070 0x70 70 70 70
The input displayed as decimal integers is:
-70 -70 56 112 56 70 112
```

```c
1  /* Fig 9.19: fig09_19.c */
2  /* Reading floating-point numbers */
3  #include <stdio.h>
4
5  /* function main begins program execution */
6  int main()
7  {
8     double a; /* define a */
9     double b; /* define b */
10    double c; /* define c */
11
12    printf( "Enter three floating-point numbers: \n" );
13    scanf( "%le%if%ig", &a, &b, &c );
14
15    printf( "Here are the numbers entered in plain\n" );
16    printf( "floating-point notation:\n" );
17    printf( "%f\n%f\n%f\n", a, b, c );
18
19    return 0; /* indicates successful termination */
20
21 } /* end main */
```

```c
1  /* Fig 9.20: fig09_20.c */
2  /* Reading characters and strings */
3  #include <stdio.h>
4
5  int main()
6  {
7     char x;      /* define x */
8     char y[ 9 ]; /* define array y */
9
10    printf( "Enter a string: " );
11    scanf( "%c%s", &x, y );
12
13    printf( "The input was:\n" );
14    printf( "the character \"%c\" ", x );
15    printf( "and the string \"%s\"\n", y );
16
17    return 0; /* indicates successful termination */
18
19 } /* end main */
```

Program Output

```
Enter a string: Sunday
The input was:
the character "S" and the string "unday"
```

```
1   /* Fig 9.21: fig09_21.c */
2   /* Using a scan set */
3   #include <stdio.h>
4
5   /* function main begins program execution */
6   int main()
7   {
8      char z[ 9 ]; /* define array z */
9
10     printf( "Enter string: " );
11     scanf( "%[aeiou]", z ); /* search for set of characters */
12
13     printf( "The input was \"%s\"\n", z );
14
15     return 0; /* indicates successful termination */
16
17  } /* end main */
```

Outline

fig09_21.c

Program Output

```
Enter string: ooeeooahah
The input was "ooeeooa"
```

```
1   /* Fig 9.22: fig09_22.c */
2   /* Using an inverted scan set */
3   #include <stdio.h>
4
5   int main()
6   {
7      char z[ 9 ] = { '\0' }; /* initialize array z */
8
9      printf( "Enter a string: " );
10     scanf( "%[^aeiou]", z ); /* inverted scan set */
11
12     printf( "The input was \"%s\"\n", z );
13
14     return 0; /* indicates successful termination */
15
16  } /* end main */
```

Outline

fig09_22.c

Program Output

```
Enter a string: String
The input was "Str"
```

```
1   /* Fig 9.23: fig09_23.c */
2   /* Inputting data with a field width */
3   #include <stdio.h>
4
5   int main()
6   {
7      int x; /* define x */
8      int y; /* define y */
9
10     printf( "Enter a six digit integer: " );
11     scanf( "%2d%d", &x, &y );
12
13     printf( "The integers input were %d and %d\n", x, y );
14
15     return 0; /* indicates successful termination */
16
17  } /* end main */
```

Outline

fig09_23.c

Program Output

```
Enter a six digit integer: 123456
The integers input were 12 and 3456
```

```
1   /* Fig 9.24: fig09_24.c */
2   /* Reading and discarding characters from the input stream */
3   #include <stdio.h>
4
5   int main()
6   {
7      int month1; /* define month1 */
8      int day1;   /* define day1 */
9      int year1;  /* define year1 */
10     int month2; /* define month2 */
11     int day2;   /* define day2 */
12     int year2;  /* define year2 */
13
14     printf( "Enter a date in the form mm-dd-yyyy: " );
15     scanf( "%d%*c%d%*c%d", &month1, &day1, &year1 );
16
17     printf( "month = %d  day = %d  year = %d\n\n", month1, day1, year1 );
18
19     printf( "Enter a date in the form mm/dd/yyyy: " );
20     scanf( "%d%*c%d%*c%d", &month2, &day2, &year2 );
21
22     printf( "month = %d  day = %d  year = %d\n", month2, day2, year2 );
23
24     return 0; /* indicates successful termination */
25
26  } /* end main */
```

Outline

fig09_24.c

```
Enter a date in the form mm-dd-yyyy: 11-18-2003
month = 11  day = 18  year = 2003

Enter a date in the form mm/dd/yyyy: 11/18/2003
month = 11  day = 18  year = 2003
```

▲ <u>Outline</u>
▼
Program Output