Input/output formattato

Formattare input e output

- La presentazione dei risultati è una parte cruciale nella soluzione di un problema
- Utilizziamo le funzionalità di formattazione di
  - `printf` invia dati allo stream standard output
  - `scanf` legge dati dallo stream standard input
- E' necessaria l'intestazione `<stdio.h>` per utilizzare queste funzioni

Stream

- Le operazioni di input e output si svolgono tramite sequenze di byte chiamate **stream:**
  - Nelle operazioni di input, i byte fluiscono da un dispositivo (es. tastiera, disco rigido) alla memoria principale
  - Nelle operazioni di output, i byte fluiscono dalla memoria principale a un dispositivo (es. schermo, stampante)
- All'inizio dell'esecuzione, un programma ha accesso a tre streams:
  - Standard input, connesso alla tastiera
  - Standard output, connesso allo schermo
  - Standard error, connesso anch'esso allo schermo
- I sistemi operativi permettono spesso di reindirizzare questi stream verso altri dispositivi

Formattazione dell'output

- `printf` utilizza una stringa di controllo del formato per definire il formato dell'output
- La stringa di controllo del formato può includere:
  - Specificatori di conversione
  - Larghezza di campo
  - Precisioni
  - Caratteri letterali
- Insieme al segno di percentuale (%), formano le specifiche di conversione

Formattazione dell'output

- Funzionalità di formattazione di `printf`:
  - Arrotondare valori in virgola mobile a un numero specifico di cifre decimali
  - Allineare colonne di numeri al punto decimale
  - Allineare l'output a destra o a sinistra
  - Inserire caratteri letterali in posizioni precise
  - Rappresentare numeri in virgola mobile in formato esponenziale
  - Rappresentare interi senza segno in formato ottale o esadecimale
  - Stampare tutti i tipi di dati con precisione e larghezza di campo fissata

Formattazione dell'output

- La funzione `printf` ha la forma:

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

- Il primo argomento descrive il formato dell'output
- Gli altri argomenti (facoltativi) corrispondono a una specifica di conversione della stringa di controllo del formato
- Ogni specifica di conversione inizia con % e termina con uno specificatore di conversione
- Ci possono essere molte specifiche di conversione in una stringa di controllo del formato

Stampa di interi

- Un intero è un numero senza punto decimale
- I numeri interi possono essere stampati secondo diversi formati descritti dai seguenti specificatori di conversione di interi

Specificatore di conversione	Descrizione
`d`	Stampa come un intero decimale con segno.
`i`	Stampa come un intero decimale con segno.
`o`	Stampa come un intero ottale senza segno.
`u`	Stampa come un intero decimale senza segno.
`x` o `X`	Stampa come un intero esadecimale senza segno. `X` usa le cifre `0–9` e le lettere maiuscole `A–F`, e `x` usa le cifre `0–9` e le lettere minuscole `a–f`.
`h`, `l` o `ll` (lettera "elle")	Questi **modificatori di lunghezza** vanno posti prima di uno specificatore di conversione di interi per indicare che il valore da stampare è di tipo intero `short`, `long` o `long long`.

Fonte: Deitel & Deitel

Stampa di interi

```c
// fig09_01.c
// Uso di specificatori di conversione di interi
#include <stdio.h>

int main(void) {
   printf("%d\n", 455);
   printf("%i\n", 455); // i come d in printf
   printf("%d\n", +455); // non viene stampato il segno piu'
   printf("%d\n", -455); // viene stampato il segno meno
   printf("%hd\n", 32000); // stampa come tipo short
    printf("%ld\n", 2000000000L); // stampa come tipo long
    printf("%o\n", 455); // ottale
    printf("%u\n", 455);
    printf("%u\n", -455);
    printf("%x\n", 455); // esadecimale con lettere minuscole
    printf("%X\n", 455); // esadecimale con lettere maiuscole
 }
```

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

Fonte: Deitel & Deitel

Stampa di numeri in virgola mobile

- I numeri in virgola mobile contengono un punto decimale
- I valori in virgola mobile possono essere stampati secondo diversi formati descritti dai seguenti specificatori di conversione

Specificatore di conversione	Descrizione
`e` o `E`	Stampa un valore in virgola mobile in notazione esponenziale.
`f` o `F`	Stampa i valori in virgola mobile nella notazione in virgola fissa.
`g` o `G`	Stampa un valore in virgola mobile o nel formato `f` oppure nel formato esponenziale `e` (o `E`), in base alla grandezza del valore.
`L`	Questo modificatore di lunghezza va posto prima dello specificatore di conversione di numeri in virgola mobile per indicare che viene stampato un valore in virgola mobile `long double`.

Fonte: Deitel & Deitel

Stampa di numeri in virgola mobile

```c
// fig09_02.c
// Uso degli specificatori di conversione di numeri in virgola mobile
#include <stdio.h>

int main(void) {
   printf("%e\n", 1234567.89);
   printf("%e\n", +1234567.89); // non viene stampato il segno piu'
   printf("%e\n", -1234567.89); // viene stampato il segno meno
   printf("%E\n", 1234567.89);
   printf("%f\n", 1234567.89); // sei cifre a destra del punto decimale
    printf("%g\n", 1234567.89); // stampa con la lettera minuscola e
    printf("%G\n", 1234567.89); // stampa con la lettera maiuscola E
 }
```

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

Fonte: Deitel & Deitel

Stampa di stringhe e caratteri

- Lo specificatore di conversione `c`:
  - è usato per stampare caratteri singoli
  - Richiede un argomento `char`
- Lo specificatore di conversione `s`:
  - è usato per stampare stringhe
  - Richiede un argomento puntatore a `char`
  - Stampa caratteri fino al carattere nullo di terminazione (`'\0'`)
  - Se la stringa non include `'\0'`, il risultato è indefinito

Stampa di stringhe e caratteri

```c
// fig09_03.c
// Uso degli specificatori di conversione di caratteri e stringhe
#include <stdio.h>

int main(void) {
   char character = 'A'; // inizializza un char
   printf("%c\n", character);

   printf("%s\n", "This is a string");

   char string[] = "This is a string"; // inizializza un array di char
   printf("%s\n", string);

    const char *stringPtr = "This is also a string"; // puntatore a char
    printf("%s\n", stringPtr);
 }
```

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

Fonte: Deitel & Deitel

Altri specificatori di conversione

- Altri possibili specificatori di conversione sono `p` e `%`
- `p` stampa un valore puntatore in un formato determinato dall'implementazione (solitamente esadecimale)
- `%` stampa il carattere di percentuale

Altri specificatori di conversione

```c
// fig09_04.c
// Uso degli specificatori di conversione p e %
#include <stdio.h>

int main(void) {
   int x = 12345;
   int *ptr = &x;

   printf("The value of ptr is %p\n", ptr);
   printf("The address of x is %p\n\n", &x);

    printf("Printing a %% in a format control string\n");
 }
```

```
The value of ptr is 0x7ffff6eb911c
The address of x is 0x7ffff6eb911c
Printing a % in a format control string
```

Fonte: Deitel & Deitel

Stampare con larghezza di campo

- La dimensione di un campo in cui sono stampati i dati è specificata dalla larghezza di campo
- Se la larghezza di campo è più grande dei dati da stampare, i dati vengono allineati a destra
- Un intero è inserito tra % e lo specificatore di conversione per definire la larghezza di campo (es, `%11d`)

Stampare con larghezza di campo

```c
// fig09_05.c
// Allineamento a destra di interi in un campo
#include <stdio.h>

int main(void) {
   printf("%4d\n", 1);
   printf("%4d\n", 12);
   printf("%4d\n", 123);
   printf("%4d\n", 1234);
   printf("%4d\n\n", 12345);

    printf("%4d\n", -1);
    printf("%4d\n", -12);
    printf("%4d\n", -123);
    printf("%4d\n", -1234);
    printf("%4d\n", -12345);
 }
```

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

Fonte: Deitel & Deitel

Stampare con precisione

- La funzione `printf` permette di specificare la precisione con cui i dati sono stampati
  - Con interi la precisione indica il numero minimo di cifre da stampare
    - Se il valore contiene meno cifre vengono aggiunti degli 0 a sinistra
  - Con numeri in virgola mobile la precisione indica il numero di cifre da stampare dopo il punto decimale
  - Con gli specificatori `g` e `G` indica il numero massimo di cifre significative da stampare
  - Con lo specificatore `s` indica il numero massimo di caratteri stampati dall'inizio della stringa

Stampare con precisione

```c
// fig09_06.c
// Stampa di interi, numeri in virgola mobile e stringhe con precisioni
#include <stdio.h>

int main(void) {
   puts("Using precision for integers");
   int i = 873; // inizializza int i
   printf("\t%.4d\n\t%.9d\n\n", i, i);

   puts("Using precision for floating-point numbers");
   double f = 123.94536; // inizializza double f
    printf("\t%.3f\n\t%.3e\n\t%.3g\n\n", f, f, f);

    puts("Using precision for strings");
    char s[] = "Happy Birthday"; // inizializza l'array di char s
    printf("\t%.11s\n", s);
 }
```

```
Using precision for integers
        0873
        000000873
Using precision for floating-point numbers
        123.945
        1.239e+02
        124
Using precision for strings
        Happy Birth
```

Fonte: Deitel & Deitel

Stampare con larghezza di campo e precisione

- E’ possibile stampare combinando entrambi i parametri
- Inserire tra il segno percentuale e lo specificatore di conversione
  - Laghezza di campo
  - Punto decimale
  - Valore di precisione

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

- Stampa

  123.457

Stampa di letterali e sequenze di escape

- Normalmente i caratteri letterali inclusi nelle stringhe di controllo del formato sono stampati da `printf`
- Alcuni caratteri sono un problema
- Ad esempio le virgolette `"` sono usate per delimitare la stringa di controllo
- Caratteri di controllo come newline e tab sono rappresentate con sequenze di escape
- Una sequenza di escape è rappresentata da un backslash (`\`) seguito da un carattere di escape

Stampa di letterali e sequenze di escape

Sequenza di escape	Descrizione
`\'` (virgoletta singola)	Stampa il carattere di virgoletta singola (`'`).
`\"` (virgolette doppie)	Stampa il carattere di virgolette doppie (`"`).
`\?` (punto interrogativo)	Stampa il carattere di punto interrogativo (`?`).
`\\` (backslash)	Stampa il carattere di backslash (`\`).
`\a` (messaggio di avviso o squillo)	Causa l'emissione di un segnale acustico o visivo (di solito con un segnale luminoso nella finestra di esecuzione del programma).
`\b` (backspace)	Sposta il cursore indietro di una posizione sulla riga corrente.
`\f` (pagina nuova o avanzamento pagina)	Sposta il cursore all'inizio della successiva pagina logica.
`\n` (newline)	Sposta il cursore all'inizio della riga *successiva*.
`\r` (ritorno a capo)	Sposta il cursore all'inizio della riga *corrente*.
`\t` (tab orizzontale)	Sposta il cursore alla posizione del tab orizzontale successivo.
`\v` (tab verticale)	Sposta il cursore alla posizione del tab verticale successivo.

Fonte: Deitel & Deitel

Intervallo

Fonte: PlaygroundAI

Input formattato

- La funzione `scanf` può essere utilizzata per formattare l'input
- `scanf` contiene una stringa di controllo del formato che descrive il formato da inserire e contiene
  - Specificatori di conversione
  - Caratteri letterali
- Le funzionalità di formattazione di `scanf` sono:
  - Leggere tutti i tipi di dati
  - Leggere caratteri specifici
  - Saltare caratteri specifici

Input formattato

- La forma di scanf è

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

- Il primo argomento descrive i formati dell'input
- Gli altri argomenti sono puntatori alle variabili nelle quali verrà memorizzato l'input
- Linee guida:
  - Quando leggete i dati, stampate un prompt di richiesta all'utente di un dato o una porzione di dati alla volta
  - Evitate di chiedere all'utente di inserire molti dati in risposta a una singola richiesta
  - Considerate sempre il funzionamento del programma quando vengono inseriti dati scorretti

Input formattato

Specificatore di conversione	Descrizione
Interi	
`d`	Legge un intero decimale con o senza segno. L'argomento corrispondente è un puntatore a un `int`.
`i`	Legge un intero decimale, ottale o esadecimale con o senza segno. L'argomento corrispondente è un puntatore a un `int`.
`o`	Legge un intero ottale. L'argomento corrispondente è un puntatore a un `int` senza segno.
`u`	Legge un intero decimale senza segno. L'argomento corrispondente è un puntatore a un `int` senza segno.
`x` o `X`	Legge un intero esadecimale. L'argomento corrispondente è un puntatore a un `int` senza segno.
`h`, `l` e `ll`	Vanno posti prima di uno qualsiasi degli specificatori di conversione di interi per indicare che si deve leggere, rispettivamente, un intero `short`, `long` o `long long`.
Numeri in virgola mobile	
`e`, `E`, `f`, `g` o `G`	Legge un valore in virgola mobile. L'argomento corrispondente è un puntatore a una variabile in virgola mobile.
`l` o `L`	Vanno posti prima di uno qualsiasi degli specificatori di conversione di numeri in virgola mobile per indicare che si deve leggere un valore `double` o `long double`. L'argomento corrispondente è un puntatore a una variabile `double` o `long double`.

Fonte: Deitel & Deitel

Input formattato

Caratteri e stringhe	
`c`	Legge un carattere. L'argomento corrispondente è un puntatore a `char`; non viene aggiunto alcun carattere nullo `('\0')`.
`s`	Legge una stringa. L'argomento corrispondente è un puntatore a un array di tipo `char`, grande abbastanza da contenere la stringa e un carattere nullo di terminazione `('\0')` che viene aggiunto automaticamente.
Insieme di scansione	
`[`*caratteri per la scansione*`]`	Esegue la scansione di una stringa per un insieme di caratteri memorizzati in un array.
Altri	
`p`	Legge un indirizzo nello stesso formato prodotto da un'istruzione `printf` con lo specificatore di formato `%p`.
`n`	Memorizza il numero di caratteri letti fino a quel punto nella chiamata corrente a `scanf`. L'argomento corrispondente è un puntatore a un `int`.
`%`	Ignora un segno di percentuale `(%)` nell'input.

Fonte: Deitel & Deitel

Leggere interi

```c
// fig09_12.c
// Lettura di input con specificatori di conversione di interi
#include <stdio.h>

int main(void) {
   int a = 0;
   int b = 0;
   int c = 0;
   int d = 0;
   int e = 0;
    int f = 0;
    int g = 0;

    puts("Enter seven integers: ");
    scanf( "%d%i%i%i%o%u%x", &a, &b, &c, &d, &e, &f, &g);

    puts("\nThe input displayed as decimal integers is:");
    printf("%d %d %d %d %d %d %d\n", a, b, c, d, e, f, g);
 }
```

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

Fonte: Deitel & Deitel

Leggere numeri in virgola mobile

```c
// fig09_13.c
// Lettura di input con specificatori di conversione di numeri in virgola mobile
#include <stdio.h>

int main(void) {
   double a = 0.0;
   double b = 0.0;
   double c = 0.0;

   puts("Enter three floating-point numbers:");
   scanf("%le%lf%lg", &a, &b, &c);

    puts("\nUser input displayed in plain floating-point notation:");
    printf("%f\n%f\n%f\n", a, b, c);
 }
```

```
Enter three floating-point numbers:
1.27987 1.27987e+03 3.38476e-06
User input displayed in plain floating-point notation:
1.279870
1279.870000
0.000003
```

Fonte: Deitel & Deitel

Leggere caratteri e stringhe

```c
// fig09_14.c
// Lettura di caratteri e stringhe
#include <stdio.h>

int main(void) {
   char x = '\0';
   char y[9] = "";

   printf("%s", "Enter a string: ");
   scanf( "%c%8s", &x, y);

    printf("The input was '%c' and \"%s\"\n", x, y);
 }
```

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

Fonte: Deitel & Deitel

Insiemi di scansione

- Insieme di scansione:
  - Insieme di caratteri racchiusi fra parentesi quadre (`[]`)
  - Preceduto da un segno percentuale nella stringa di controllo del formato
- Legge solo caratteri contenuti nell'insieme di scansione
- La lettura termina quando `scanf` incontra un carattere non contenuto nell'insieme di scansione
- Se il primo carattere nello stream di input non è nell'insieme di scansione, `scanf` non modifica il suo argomento array corrispondente

Fonte: Deitel & Deitel

Insiemi di scansione

```c
// fig09_15.c
// Uso di un insieme di scansione
#include <stdio.h>

int main(void) {
   char z[9] = "";

   printf("%s", "Enter string: ");
   scanf("%8[aeiou]", z); // cerca un insieme di caratteri

   printf("The input was \"%s\"\n", z);
}
```

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

Fonte: Deitel & Deitel

Insiemi di scansione invertito

- Insieme di scansione invertito:
  - Legge solo caratteri non contenuti nell'insieme di scansione
- Per creare un insieme di scansione invertito, mettere un accento circonflesso (^) prima dei carattere nell'insieme di scansione

```c
// fig09_16.c
// Uso di un insieme di scansione invertito
#include <stdio.h>

int main(void) {
   char z[9] = "";

   printf("%s", "Enter a string: ");
   scanf("%8[^aeiou]", z); // insieme di scansione invertito

   printf("The input was \"%s\"\n", z);
}
```

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

Fonte: Deitel & Deitel

Scansione con larghezza di campo

- Leggere un numero specifico di caratteri dallo stream di input

```c
// fig09_17.c
// Lettura di dati con una larghezza di campo
#include <stdio.h>

int main(void) {
   int x = 0;
   int y = 0;

   printf("%s", "Enter a six digit integer: ");
   scanf( "%2d%d", &x, &y);

    printf("The integers input were %d and %d\n", x, y);
 }
```

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

Fonte: Deitel & Deitel

Tralasciare caratteri

- Può essere necessario tralasciare dei caratteri
- Es: `11-10-1999`
- I numeri della data devono essere memorizzati
- I trattini devono essere scartati
- Includere i caratteri da scartare

```
scanf("%d-%d-%d", &month, &day, &year);
```

Carattere di soppressione

- Il carattere * di soppressione permette a `scanf` di leggere qualsiasi tipo di dato in input e scartarlo

```c
// fig09_18.c
// Lettura ed eliminazione di caratteri dallo stream di input
#include <stdio.h>

int main(void) {
   int month = 0;
   int day = 0;
   int year = 0;
   printf("%s", "Enter a date in the form mm-dd-yyyy: ");
   scanf( "%d%*c%d%*c%d", &month, &day, &year);
    printf("month = %d  day = %d  year = %d\n\n", month, day, year);

    printf("%s", "Enter a date in the form mm/dd/yyyy: ");
    scanf( "%d%*c%d%*c%d", &month, &day, &year);
    printf("month = %d  day = %d  year = %d\n", month, day, year);
 }
```

```
Enter a date in the form mm-dd-yyyy: 07-04-2021
month = 7  day = 4  year = 2021
Enter a date in the form mm/dd/yyyy: 01/01/2021
month = 1  day = 1  year = 2021
```

Fonte: Deitel & Deitel

Recap

- Le operazioni di input e output vengono effettuate tramite stream, che sono sequenze di byte
- Standard input connesso alla tastiera
- Standard output e error allo schermo
- Stringa di controllo del formato definisce i formati in cui vengono stampati i valori di output e include:
  - Specificatori di conversione
  - Larghezza di campo
  - Precisione
  - Caratteri