

Comandi di Input/Output in Matlab

Il comando `format` controlla la configurazione numerica dei valori esposta da MATLAB; il comando regola solamente come i numeri sono **visualizzati** o **stampati**, non come MATLAB li calcola o li salva in memoria (MATLAB lavora sempre in doppia precisione). Di seguito sono indicati diversi formati, insieme con l'output che ne deriva, prodotto da un vettore `x` con componenti di dimensioni diverse.

```
>> x = [4/3 1.2345e-6];
>> format short
>> x
    1.3333 0.0000
>> format short e
>> x
    1.3333e+000 1.2345e-006
>> format short g
>> x
    1.3333 1.2345e-006
>> format long
>> x
    1.33333333333333 0.00000123450000
>> format long e
>> x
    1.33333333333333e+000 1.234500000000000e-006
>> format long g
>> x
    1.33333333333333 1.2345e-006
```

Se l'utente vuole definire il formato di uscita in modo *personalizzato*, deve usare le funzioni `sprintf` e `fprintf` con la definizione dei formati.

Nella tabella che segue sono riportati alcuni codici di formato con il relativo significato:

Codice di formato	Significato
<code>%s</code>	formato stringa
<code>%d</code>	formato decimale
<code>%g</code>	seleziona il formato per numeri interi, fixed point o esponenziali
<code>%f</code>	formato fixed point del numero (esempio 1343.675432)
<code>%e</code>	formato esponenziale del numero (esempio 1.34376e+003)
<code>\n</code>	inserisce carattere di ritorno a capo
<code>\t</code>	inserisce carattere di tabulazione

Esempio di alcuni formati utilizzabili con `fprintf` e `sprintf`

Valore	%6.3f	%6.0f	%6.3e	%6.3g	%6.3d	%6.0d
2	2.000	2	2.000e+000	2	002	2
0.02	0.020	0	2.000e-002	0.02	2.000e-002	2e-002
200	200.000	200	2.000e+002	200	200	200
$\sqrt{2}$	1.414	1	1.414e+000	1.41	1.414e+000	1e+000
$\sqrt{0.02}$	0.141	0	1.414e-001	0.141	1.414e-001	1e-001
$\sqrt{2 \cdot 10^{-20}}$	0.000	0	1.414e-010	1.41e-010	1.414e-010	1e-010

Gestione dell'output su video

`disp ('stringa')` consente di visualizzare sullo schermo la sequenza di caratteri *stringa*.

Esempi:

```
>> st = 'Questa e' una stringa';
>> disp (st);
Questa e' una stringa
>> disp ('Questa e' una seconda stringa');
Questa e' una seconda stringa
```

Per visualizzare un insieme di dati di output con un certo *formato* si utilizzano i comandi `fprintf` e `sprintf`:

`fprintf ('formato', variabili)`

scrive su video il valore delle *variabili* indicate, utilizzando il *formato* definito.

Il *formato* è una stringa che contiene i caratteri che si vogliono visualizzare e, nelle posizioni in cui si vuole venga inserito il valore, deve essere indicato uno dei formati preceduti dal carattere %. Tali codici di formati sono abitualmente seguiti da due interi separati da un punto (ad esempio 6.3). Il primo numero indica quante colonne si desiderano impegnare in uscita ed il secondo il numero di cifre della parte frazionaria. I formati \n e \t servono per organizzare le linee dell'output.

`str = sprintf ('formato', variabili)`

indirizza su una stringa di testo di nome *str* le *variabili* indicate, con il *formato* definito. Per visualizzare il tutto, è sufficiente poi utilizzare in comando `disp(str)`.

Esempi:

```
>> n=4; x=1234.5467; err=1.345677e-4;
>> fprintf('Iter = %5.0f \t Valore x = %10.7f \t Errore = %7.2e', ...
           n, x, err);
Iter =      4    Valore x = 1234.5467000    Errore = 1.35e-004
>>
>> st=sprintf('Iter = %5.0f \t Valore x = %10.7f \t Errore = %7.2e', ...
               n, x, err);
>> disp(st);
Iter =      4    Valore x = 1234.5467000    Errore = 1.35e-004
```

Gestione dell'output su file

Per scrivere su file un insieme di dati di output con un certo *formato* si utilizzano i comandi **fopen**, **fprintf** e **fclose**:

fid = fopen ('stringa', 'w')

dove *fid* è una variabile che identifica il file e '*stringa*' definisce il nome del file, apre il file in scrittura (parametro '*w*').¹

fprintf (fid, 'formato', variabili)

dove *fid* è l'identificatore del file di uscita, scrive nel file *fid* il valore delle *variabili* con il *formato* assegnato.

fclose (fid)

dove *fid* è la variabile che identifica il file, chiude il file.

Esempi:

```
>> n=4; x=1234.5467; err=1.345677e-4;
>> f1 = fopen('FILEOUT1.DAT','w');
>> fprintf(f1,'Iter = %5.0f \t Valore x = %10.7f \t Errore = %7.2e', ...
           n, x, err);
>> fclose(f1);
>> type FILEOUT1.DAT
```

```
Iter =      4      Valore x = 1234.5467000      Errore = 1.35e-004
>>
>> A = [1 2 3; 4 5 6; 7 8 9];
>> f2 = fopen('FILEOUT2.DAT','w');
>> fprintf(f2,'%5.0f', A);
>> fclose(f2);
>> type FILEOUT2.DAT
```

```
    1      4      7      2      5      8      3      6      9
>>
>> f3 = fopen('FILEOUT3.DAT','w');
>> fprintf(f3,'%5.0f \n', A);
>> fclose(f3);
>> type FILEOUT3.DAT
    1
    4
    7
    2
    5
    8
    3
    6
    9
```

¹Se si lavora in ambiente Windows, è consigliabile usare il parametro '**wt**' (*scrivi in modalità testo*) al posto di '**w**', per evitare possibili incompatibilità con le applicazioni di lettura files di testo di Windows.

```

>> f4 = fopen('FILEOUT4.DAT', 'w');
>> for i=1:3
    fprintf(f4, '%5.0f %5.0f %5.0f \n', A(i,:));
end;
>> fclose(f4);
>> type FILEOUT4.DAT

1      2      3
4      5      6
7      8      9

>> x = 0::1:1;
>> y = [x; exp(x)];
>> f5 = fopen('exp.txt', 'w');
>> fprintf(f5, '%6.2f %12.8f\n', y);
>> fclose(f5)
>> type exp.txt

0.00  1.00000000
0.10  1.10517092
0.20  1.22140276
0.30  1.34985881
0.40  1.49182470
0.50  1.64872127
0.60  1.82211880
0.70  2.01375271
0.80  2.22554093
0.90  2.45960311
1.00  2.71828183

```

Esempi di scrittura su file di matrici di dimensioni $n \times m$:

```

>> A=[1 4 6 5 8; 6 7 4 3 1; 8 7 3 2 1];
>> [n,m] = size(A);
>> f6 = fopen('matrice.out', 'w');
>> for i=1:n
>>     for j=1:m
>>         fprintf(f6, '%8.4f\t', A(i,j));
>>     end
>>     fprintf(f6, '\n');
>> end
>> fclose(f6);
>> type matrice.out

1.0000      4.0000      6.0000      5.0000      8.0000
6.0000      7.0000      4.0000      3.0000      1.0000
8.0000      7.0000      3.0000      2.0000      1.0000

```

```

>> A=rand(3,2);
>> [n,m] = size(A);
>> f7 = fopen('randmatrix.out','w');
>> fprintf(f7,'%s \n', ' Scrive Matrice Random');
>> fprintf(f7,'\n');
>> for i=1:n
>>     for j=1:m
>>         fprintf(f6,'%18.16e\t',A(i,j));
>>     end
>>     fprintf(f6,'\n');
>> end
>> fclose(f6);
>> A =

```

0.0503	0.8744
0.4154	0.0150
0.3050	0.7680

```

>> type randmatrix.out
    Scrive Matrice Random

5.0268803746872939e-002    8.7436717158762622e-001
4.1537486044322269e-001    1.5009498676616266e-002
3.0499867700349187e-001    7.6795039001114140e-001

```

Gestione dell'input da tastiera

`variabile = input ('stringa')`

attende un dato in ingresso da tastiera, dopo aver visualizzato la stringa di caratteri *stringa*, e lo assegna a `variabile`. Utilizzabile per assegnare una sola variabile.

Esempi:

```

>> nmax = input('Dammi il numero massimo di iterazioni ')
Dammi il numero massimo di iterazioni 40
nmax =
40

>> x01 = input('Dammi i due valori iniziali ')
Dammi i due valori iniziali [0.0 2.0]
x01 =
    0      2
>> x0 = x01(1);
>> x1 = x01(2);

```

Gestione dell'input da file

Per leggere da un file un insieme di dati si utilizzano i comandi `fopen`, `fscanf` e `fclose`:

`fid = fopen ('stringa')`

dove `fid` è una variabile che identifica il file e '`stringa`' definisce il nome del file, apre il file.

`var = fscanf (fid, formato, size)`

dove `fid` è la variabile che identifica il file, legge tutti i dati contenuti nel file identificato da `fid`, convertendoli in base al `formato` specificato e memorizzandoli nella variabile `var`. `size` indica la dimensione della variabile `var` e può essere scritto come

`nval` legge `nval` numeri memorizzandoli in un vettore colonna

`[nrighe, ncol]` legge `nrighe*ncol` numeri memorizzandoli in una matrice
che ha tale dimensione

Il valore `ncol` può essere posto uguale alla costante `inf`. In tal caso legge per righe tutti i numeri del file.

`fclose (fid)`

dove `fid` è la variabile che identifica il file, chiude il file.

Esempio 1:

```
>> type dati.in
0.10    2
0.20    3
0.30    4

>> f1 = fopen('dati.in');
>> vet = fscanf(f1,'%g %g',2) % legge la prima riga
vet =
    0.1000
    2.0000
>> vet1 = fscanf(f1,'%g %g',2) % legge la seconda riga
vet1 =
    0.2000
    3.0000
>> vet2 = fscanf(f1,'%g %g',2) % legge la terza riga
vet2 =
    0.3000
    4.0000
>> fclose(f1)
```

Esempio 2:

```
>> type exp.txt
 0.00  1.00000000
 0.10  1.10517092
 0.20  1.22140276
 0.30  1.34985881
 0.40  1.49182470
 0.50  1.64872127
 0.60  1.82211880
 0.70  2.01375271
 0.80  2.22554093
 0.90  2.45960311
 1.00  2.71828183

>> f1 = fopen('exp.txt');
>> B = fscanf(f1,'%g %g',[2 inf]); % il file ha 2 colonne
>> % legge per righe 2 elementi sino alla fine del file
>> % B sara' una matrice di dimensione 2 righe x 11 colonne
>> fclose(f1)
>> % bisogna trasporre la matrice per avere 11 righe e 2 colonne
>> B = B'
B =
      0          1
      0.1        1.10517092
      0.2        1.22140276
      0.3        1.34985881
      0.4        1.4918247
      0.5        1.64872127
      0.6        1.8221188
      0.7        2.01375271
      0.8        2.22554093
      0.9        2.45960311
      1          2.71828183
```

Esempio 3:

```
>> type FILEOUT4.DAT % contiene una matrice scritta per righe
    1      2      3
    4      5      6
    7      8      9
>> f4 = fopen('FILEOUT4.DAT');
>> A = fscanf(f4,'%g %g %g',[3 inf]); % il file ha tre colonne
>> % legge per righe 3 elementi sino alla fine del file
>> % A sara' una matrice di dimensione 3 righe x 3 colonne
```

```

>> fclose(f4)
>> % bisogna trasporre la matrice per avere le componenti
>> % corrispondenti a quelle della matrice del file
>> A = A'
A =
    1     2     3
    4     5     6
    7     8     9

```

Il comando `fscanf` può anche essere utilizzato nella forma

$$[var, numval] = \text{fscanf} (fid, formato, size)$$

dove il valore restituito `numval` indica il numero di dati letti sull'intero file.

Esempio 4:

```

>> type exp.txt
0.00    1.00000000
0.10    1.10517092
0.20    1.22140276
0.30    1.34985881
0.40    1.49182470
0.50    1.64872127
0.60    1.82211880
0.70    2.01375271
0.80    2.22554093
0.90    2.45960311
1.00    2.71828183
>> f1 = fopen('exp.txt');
>> [B, nn] = fscanf(f1, '%g %g', [2 inf]); % ha due colonne
>> fclose(f1)
>> nn
nn =
    22
>> % ha letto 22 dati!

```

Esempio di script

```
% tabella.m
% costruisce la tabella dei valori che il seno e il coseno
% assumono in n punti equispaziati di (0,pi)
n=input('Inserisci il numero di valori: ');
x=linspace(0,pi,n);
s=sin(x);
c=cos(x);
disp('-----');
fprintf('k\t x\t sin x\t cos x\n');
disp('-----');
for i=1:n
    fprintf('%d\t %3.2f\t %8.5f\t %8.5f\n',i,x(i),s(i),c(i));
end
```

Memorizzando su file tale script e facendolo eseguire si ottiene una tabella:

```
>> tabella
>> Inserisci il numero di valori: 5
-----
k      x      sin x      cos x
-----
1      0.00    0.00000   1.00000
2      0.79    0.70711   0.70711
3      1.57    1.00000   0.00000
4      2.36    0.70711  -0.70711
5      3.14    0.00000  -1.00000
```

Se si vuole utilizzare il comando sprintf in luogo di fprintf nello script dell'esempio precedente, si deve scrivere

```
% tabella.m
% costruisce la tabella dei valori che il seno e il coseno
% assumono in n punti equispaziati di (0,pi)
n=input('Inserisci il numero di valori: ');
x=linspace(0,pi,n);
s=sin(x);
c=cos(x);
disp('-----');
stringa=sprintf('k\t x\t sin x\t cos x\n');
disp(stringa);
disp('-----');
stringa=sprintf('%d\t %3.2f\t %8.5f\t %8.5f\n',[1:n;x;s;c]);
disp(stringa)
```