Università degli Studi di Ferrara

Corso di Laurea in Matematica - A.A. 2018 - 2019

Programmazione
Lezione 21 – Input Output

Docente: Michele Ferrari - michele.ferrari@unife.it

Nelle lezioni precedenti

- Operatori relazionali e logici
- Costrutto if
- Ciclo for
- Ciclo while

In questa lezione

- Costrutto elseif
- Costrutto switch
- Funzioni per Input Output
- Input Output su e da file

Costrutto elseif

```
if Condizione1
    blocco di istruzioni
elseif Condizione2
    blocco di istruzioni
...
else
    blocco di istruzioni
end
```

Il primo blocco di istruzioni sarà eseguito solo se la Condizione1 risulta essere verificata, il secondo solo se la Condizione1 risulta essere **falsa** e la Condizione2 **vera** e così via.

Il blocco di istruzioni che segue **else** sarà eseguito soltanto se nessuna delle precedenti condizioni risulta essere vera.

Il costrutto switch

Il costrutto **switch** risulta particolarmente utile qualora si debbano eseguire numerose scelte di tipo esclusivo (se una è verificata le altre sono false)

```
switch Espressione
  case Valore1
    blocco di istruzioni
  case Valore2
    blocco di istruzioni
  ...
  otherwise
    blocco di istruzioni
end
```

I blocchi di istruzioni sono eseguiti solo se l'Espressione assume il corrispondente Valore.

L'ultimo blocco di istruzioni, indicato dalla keyword otherwise, sarà eseguito solo nel caso in cui Espressione non abbia assunto nessuno dei precedenti valori

Input/Output: disp

L'istruzione **disp** consente di visualizzare una stringa di testo sullo schermo del calcolatore.

La sintassi della funzione disp è la seguente

```
disp(stringa di caratteri)
```

dove stringa di caratteri rappresenta un array di stringhe di tipo vettore o matrice, dove ogni stringa è racchiusa tra apici.

Ad esempio potremo scrivere

```
>> disp('Oggi e'' lunedi'''')
oggi e' lunedi'
```

Osserviamo che per utilizzare i simboli di apostrofo/accento all'interno della stringa si deve utilizzare il simbolo '' come delimitatore per evitare conflitti con il segnale di inizio e fine della stringa.

Input/Output: disp

```
>> disp(['Gennaio ','Febbraio ','Marzo'])
Gennaio Febbraio Marzo
```

In questo caso l'argomento è un vettore riga di stringhe, che vengono così concatenate.

Se vogliamo costruire vettori colonna è importante ricordarsi che le stringhe devono avere tutte la stessa dimensione

```
>> disp(['Gennaio ';'Febbraio';'Marzo   '])
Gennaio
Febbraio
Marzo
```

cosa che può essere facilmente ottenuta inserendo un opportuno numero di spazi.

Input/Output: valori numerici

In molte circostanze si ha inoltre la necessità di rappresentare valori numerici in uscita (con una certa formattazione).

Vediamo due differenti possibilità, la prima tramite il seguente esempio:

```
>> x=10;
>> stringa = ['x = ', num2str(x)];
>> disp(stringa)
x = 10
```

In questo caso abbiamo utilizzato la funzione num2str che consente di convertire una variabile numerica in una stringa.

Abbiamo poi **concatenato** i due elementi stringa per ottenere un output più elaborato

Input/Output: valori numerici

Un modo più versatile di costruire stringhe di caratteri per l'output di testo con formato è fornito dalle funzioni fprintf e sprintf

Ricordano niente?

Come il nome lascia intuire sono funzioni mutuate direttamente dal linguaggio C.

La sintassi della prima è del tipo

```
fprintf(Fid, Formato, Variabili)
```

Dove

- Formato è una stringa di testo che tramite l'uso di caratteri speciali indica il tipo di formato dell'output
- Variabili è una lista opzionale di variabili separate da una virgola e che hanno un corrispondente all'interno della stringa Formato
- Fid è un identificatore opzionale del file al quale l'output è inviato

Input/Output: valori numerici

La funzione sprintf ha invece la sintassi

```
Stringa = sprintf(Formato, Variabili)
```

dove in questo caso l'output viene indirizzato su una stringa di testo.

Differenza tra sprintf e fprintf

```
>> x=10;y=5.5;

>> fprintf('x = %d e y= %f\n',x,y);
    x = 10 e y= 5.500000

>> stringa=sprintf('x = %d e y= %f\n',x,y);
>> disp(stringa)
    x = 10 e y= 5.500000
```

Input/Output: formattatori

La stringa di Formato contiene codici che specificano il tipo di variabile che deve essere convertita in stringa e rappresentata sullo schermo del calcolatore o su file.

Come per il C:

- il formato **%d** serve a visualizzare un numero intero
- il formato **%f** è utilizzato per i numeri reali

Input/Output: formattatori

Formato	Significato
%s	Formato stringa
%d	Formato senza parte frazionaria (intero)
%f	Formato numero deciamale
%e	Formato in notazione scientifica
%g	Formato in forma compatta usando %f o %e
\n	Inserisce un carattere di ritorno a capo
\t	Inserisce un carattere di tabulazione

Input/Output: formattatori - Esempio

Valore	%6.3f	%6.3e	%6d
2	2.000	2.000w+00	2
0.02	0.020	2.000e-02	2.000000e-02
200	200.000	2.000e+02	200
sqrt(2)	1.414	1.414e+00	1.414214e+00
sqrt(0.02)	0.141	1.414e-01	1.414214e-01

Input/Output: Utilizzo vettoriale

La principale differenza tra le funzioni MATLAB fprintf e sprintf e le equivalenti versioni in C è data dalla possibilità **di utilizzarle in modo vettoriale**

Abbiamo già visto in precedenza come costruire una semplice tabella di valori per le funzioni seno e coseno.

Proviamo a costruire un nuovo script che realizzi la stessa tabella di valori ma visualizzandola in modo più efficace con l'utilizzo **vettoriale** delle funzioni appena viste

Input/Output: Utilizzo vettoriale

```
% tabcossin.m
% Realizza una tabella di valori di coseno e seno
%
n=input('Inserisci il numero di valori ? : ');
x=linspace(0,pi,n);
c=cos(x);
s=sin(x);
disp('-----------------------------------------');
fprintf('k\t x(k)\t cos(x(k))\t sin(x(k))\n');
disp('-----------------------------------------');
fprintf('%d\t %3.2f\t %6.5f\t %6.5f\n',[1:n;x;c;s]);
```

Input/Output: Utilizzo vettoriale

che produce il seguente output

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

Input/Output: Utilizzo vettoriale

Si noti l’uso vettoriale di **fprintf**:

bisogna prestare attenzione nel caso in cui si utilizzi una matrice come variabile in output

- il comando legge la matrice per colonne, applicando ad ogni elemento il corrispondente descrittore di formato
- la visualizzazione avviene naturalmente per righe e questo porta ad una sorta di ’trasposizione’ della matrice

Input/Output: la funzione input

Abbiamo già trovato la funzione input in diversi esempi ed esercitazioni, vediamone le caratteristiche

La sintassi della funzione è

```
Variabile=input(Stringa di caratteri)
```

La funzione attende un ingresso da tastiera di un'**espressione MATLAB** da assegnare a Variabile.

Il valore assegnato potrà quindi essere di tipo scalare, vettoriale, oppure matrice **utilizzando la sintassi standard di MATLAB.**

Input/Output su file: il comando diary

Se volessimo salvare su file una copia della tabella generata con l'algoritmo appena visto, il modo più semplice consiste nell'utilizzo del comando **diary**

Il comando trasferisce su file tutto ciò che compare nella finestra di comando di MATLAB.

La sintassi del comando è

```
diary Nomefile
```

il comando attiverà la copia su file di testo Nomefile di tutto ciò che comparirà sulla **command window** fino a quando non verrà dato il comando

```
diary off
```

Il quale andrà ad interrompere il salvataggio.

Input/Output su file - esempio

Il seguente script costruisce la tabella visualizzata in precedenza in un file di testo chiamato tabella.txt

```
% salvatabella.m
% Esempio di uso della funzione diary
%
diary tabella.txt
tabcossin
diary off
```

E' importante notare che il comando diary, registrando la command window, andrà a salvare tutta la sessione di lavoro.

Per un maggiore controllo è possibile utilizzare il comando **save**

Input/Output su file: il comando save

Il comando save consente di registrare alcune variabili presenti nella memoria di MATLAB su file secondo la sintassi

save Nomefile ElencoVariabili Formato

dove Formato è un parametro opzionale.

Se tale parametro è omesso il file è salvato in **formato binario** e qualora il file non abbia estensione a questo è aggiunta l'estensione ".mat"

Il formato **-ascii** consente di salvare file in modalità testo

Input/Output su file: il comando save - esempio

```
% salvavalori.m
% Esempio di output su file con save
%
n=input('Inserisci il numero di valori: ');
x=linspace(0,pi,n);
c=cos(x);
s=sin(x);
z=1:n;
save tabella.dat z x c s -ascii
```

Input/Output su file: il comando load

Possiamo caricare il file precedentemente creato con il comando save utilizzando il comando **load** per leggere il file.

Sintassi:

```
load Nomefile Formato
```

Se vogliamo leggere un file di testo che non ha estensione allora l'uso del formato **-ascii** è obbligatorio anche in lettura, altrimenti è opzionale.

Utilizzando questo comando MATLAB crea una matrice contenente i valori precedentemente registrati ed a questa assegna il nome del file in lettura.

Nell'esempio precedente l'istruzione **load tabella.dat** legge i valori presenti nel file tabella.dat e li assegna ad una matrice chiamata tabella

Input/Output su file: il comando load - esempio

```
% visualizzavalori.m
% Esempio di input da file con load
%
load tabella.dat
A=tabella;
disp('-----------------------------------');
fprintf('k\t x(k)\t cos(x(k))\t sin(x(k))\n');
disp('-----------------------------------');
fprintf('%d\t %3.2f\t %6.5f\t %6.5f\n',A);
```

Input/Output su file: il comando print

In MATLAB è possibile salvare i grafici prodotti su file utilizzando diversi formati grafici.

La sintassi base del comando è la seguente

print Opzioni Nomefile

E consente di salvare il contenuto della attuale finestra grafica sul file Nomefile utilizzando un particolare formato grafico specificato in Opzioni

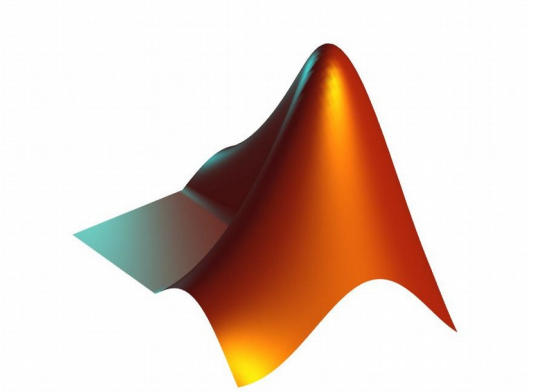
Input/Output su file: il comando print

Opzione	Formato corrispondente
-dps	PostScript bianco e nero
-deps	Encapsulated PostScript bianco e nero
-dpsc	PostScript a colori
-depsc	Encapsulated PostScript a colori
-dgif	Immagine GIF
-dbmp	Immagine Bitmap
-djpeg	Immagine JPEG compressa

Input/Output su file: il comando print - esempio

Il seguente script realizza la figura tridimensionale che MATLAB utilizza come logo, tramite il comando logo, e la salva in formato jpeg nel file mlogo.jpg

```
% salvalogo.m
% Esempio di output grafico su file
logo
print -djpeg mlogo.jpg
```



Esercizio

- Realizzare uno script che calcoli la tabella di valori di seno e coseno e la salvi su file
- Realizzare uno script che carichi il file precedentemente salvato e realizzi il grafico sovrapposto delle funzioni, si salvi poi il grafico in formato jpeg

Grazie per l’attenzione

Riferimenti

Il corso di programmazione per il primo anno della Laurea Triennale in Matematica nasce con l'intento di unire ai principi di programmazione una conoscenza basilare di uno degli strumenti software più diffusi nell'ambito matematico: Matlab.

Per la parte introduttiva di MATLAB:

L. Pareschi, G. Dimarco “Introduzione a MATLAB”, corso di Laboratorio di Calcolo Numerico 2006