Calcolo numerico e **programmazione**
Introduzione a Scilab (3)

Tullio Facchinetti
<tullio.facchinetti@unipv.it>

3 maggio 2012

23:58

http://robot.unipv.it/toolleeo

Cos'è una funzione

le funzioni sono insiemi di istruzioni che realizzano una specifica funzionalità

- utile per scrivere insiemi di istruzioni da richiamare più volte in caso di necessità
- una funzione si scrive (e se ne verifica la correttezza, oltre ad ottimizzarla) una volta sola

Componenti di una funzione

una funzione è caratterizzata da:

- il nome, che viene utilizzato per identificare univocamente la funzione e viene usato per richiamarla
- un elenco di parametri da passare alla funzione, necessari al suo corretto funzionamento
- le istruzioni che compongono la funzione
- i valori di ritorno, ovvero i risultati dell'elaborazione

Funzioni di Scilab

sono parecchie le funzioni già viste e utilizzate,
come `cos`, `log10` o `length`

- Scilab fornisce molte funzioni per gli usi più disparati
- le funzioni sono raccolte in librerie (libreria statistica, libreria di input/output, ecc.)
- le librerie sono collezioni di funzioni per usi specifici

Parametri e valori di ritorno

funzione che accetta un parametro e restituisce un valore:

```
risultato = mia_funzione(dato)
```

funzione che accetta `m` parametri e restituisce `n` valori:

```
[o1, o2, ..., on] = mia_funzione(i1, i2, ..., im)
```

parametri e valori di ritorno sono separati da virgole

Esempi

funzione che accetta un parametro e restituisce un valore:

```
y = cos(x)
```

funzione che accetta 2 parametri e restituisce 2 valori:

```
[row, col] = size(A, "*")
```

Realizzazione di funzioni

è possibile realizzare proprie funzioni per lo svolgimento di operazioni specifiche

```
function y = doppio (x)
    y = 2 * x
endfunction
```

- l'intestazione (o *header*) della funzione è `function y = doppio (x)`
- il corpo (o *body*) è, in questo caso, composto da una sola istruzione: `y = 2 * x`
- si noti come nella funzione viene assegnato il valore della variabile di output `y` (è indispensabile farlo)

Inserimento di funzioni definite dall'utente

ci sono vari modi per farlo

1. inserire da console le linee di codice che compongono la funzione (scomodo)
2. se le istruzioni che compongono la funzione sono memorizzate in un file
   - possono essere copiate e incollate nella console
   - è possibile eseguire lo script come un normale programma, in pratica "inserendo" la funzione nell'ambiente di Scilab

Librerie di funzioni

è possibile creare delle collezioni di proprie funzioni, costituendo una libreria

- una libreria è un file `binario` che contiene il codice di una o più funzioni
- una libreria si crea a partire dal file sorgente utilizzanto l'istruzione `genlib`
- le librerie sono file con estensione `.sci` (utile ma non obbligatorio)
- il nome del file `.sci` deve essere uguale al nome della prima funzione nel file
- una libreria, e quindi tutte le funzioni che ne fanno parte, vengono caricate in Scilab per mezzo della funzione `lib`
- è possibile usare direttamente solo la prima funzione presente nel file

Ritorno di zero o più valori

si possono specificare zero o più valori di ritorno

funzione che restituisce due valori di ritorno:

```
function [ y1 , y2 ] = fsemplice ( x1 , x2 )
    y1 = 2 * x1
    y2 = 3 * x2
endfunction
```

può essere richiamata in vari modi, a seconda del numero di valori di ritorno che si desidera assegnare esplicitamente

Gestione dei valori di ritorno

- nessun assegnamento
- il primo valore ritornato è assegnato ad `ans`:

```
-->fsemplice(1, 10)
 ans  =
    2.
```

Gestione dei valori di ritorno

- un solo valore assegnato
- è il primo valore restituito

```
-->y1 = fsemplice(1, 10)
 y1  =
    2.
```

Gestione dei valori di ritorno

- entrambi i valori di ritorno sono assegnati esplicitamente

```
-->[y1, y2] = fsemplice(1, 10)
 y2  =
    30.
 y1  =
    2.
```

Istruzione return

quando viene incontrata, serve per "uscire" (ritornare) immediatamente dalla funzione nella quale è stata chiamata

```
function y = mysum ( istart , iend )
  if ( istart < 0 ) then
    y = 0
    return
  end
  if ( iend < istart ) then
    y = 0
    return
  end
  y = sum ( istart : iend )
endfunction
```

Comandi utili

la verifica della correttezza del funzionamento di una funzione viene fatta per mezzo dei comandi `pause`, `abort` e `resume`

esempio di funzione errata poichè passa un parametro di troppo alla funzione `sum`:

```
function y = somma ( istart , iend )
  y = sum ( iend : istart , "parametro" )
endfunction
```

Errori di esecuzione in una funzione

l'esecuzione comporta la seguente segnalazione di errore:

```
-->somma(1, 10)
 !--error 44
Wrong argument 2.

at line       2 of function somma called by :
somma(1, 10)
```

- con `abort` si termina l'esecuzione della funzione nella quale ci troviamo
- con `resume` si continua l'esecuzione fino al termine della funzione o finchè non viene incontrata un'altra istruzione `pause`

Il comando pause

si inserisca l'istruzione `pause` nella funzione:

```
function y = somma ( istart , iend )
  pause
  y = sum ( iend : istart , "parametro" )
endfunction
```

eseguendo la funzione il prompt si ferma in questo modo

```
-->somma(1, 10)
Type 'resume' or 'abort' to return to standard level prompt
-1->
```

l'"uno" indica che ci troviamo all'interno del primo livello di funzione chiamata

Debug di una funzione in pausa

all'interno della funzione è possibile:

- visualizzare il contenuto di una variabile
- digitare qualsiasi istruzione Scilab utile a capire il funzionamento del programma

una volta verificato il corretto funzionamento della funzione andranno rimosse le istruzioni `pause` inserite per il debugging

I comandi per la generazione di grafici

i comandi possono essere impartiti:

```
plot       2D plot
surf       3D plot
contour    contour plot
pie        pie chart
histplot   histogram
bar        bar chart
barh       horizontal bar chart
hist3d     3D histogram
polarplot  plot polar coordinates
Matplot    2D plot of a matrix using colors
Sgrayplot  smooth 2D plot of a surface using colors
grayplot   2D plot of a surface using colors
```

Esempio di visualizzazione 2D

esempio di visualizzazione del grafico della funzione `sin`

```
-->x = linspace(-10,10);
-->y = sin(x);
-->plot(x, y)
```

- con `linspace` viene generato un vettore di numeri equispaziati nell'intervallo $[-10, 10]$
- il vettore `y` conterrà i valori di `sin(x)` per ciascun valore di `x`
- il comando `plot` visualizza il relativo grafico

Aggiunta di dettagli al grafico

x

Visualizzazione di una funzione definita dall'utente

x

Esportazione di un grafico

l'esportazione di un grafico consente di salvare l'immagine nel formato desiderato per il successivo utilizzo all'interno di altri programm (es., per includerlo in una presentazione)

Esportazione di un grafico

Formato vettoriale

```
xs2png    export into PNG
xs2pdf    export into PDF
xs2svg    export into SVG
xs2eps    export into Encapsulated Postscript
xs2ps     export into Postscript
xs2emf    export into EMF (only for Windows)
```

Formato bitmap

```
xs2fig    export into FIG
xs2gif    export into GIF
xs2jpg    export into JPG
xs2bmp    export into BMP
xs2ppm    export into PPM
```