Programmare in Scilab: Funzioni

Corso di Informatica

CdL: Chimica

Claudia d'Amato
claudia.damato@di.uniba.it

Le funzioni…

- **Intestazione di funzione memorizzata in un file**
  - function [y1, y2, …, yn] = miafunzione(x1, x2, …, xm)

    xi sono i paramteri di input

    yi sono i valori restituiti dalla funzione

    la funzione termina quando si raggiunge la parola chiave **endfunction**
- **Scilab non fa differenza tra procedura e funzione**
  - **Una procedura sarà sempre dichiarata con la parola chiave function ma non ritornerà alcun valore**

...Le funzioni

- **Le variabili dichiarata all'interno della funzione sono variabili locali ad essa** (a differenza degli script)
  - **Attenzione:** Se una variabile non è definita nella funzione e non è uno dei parametri di input ma ha lo stesso nome di una variabile globale, il valore della variabile globale sarà considerato nella funzione
  - I cambiamenti della variabile sono comunque locali alla funzione, ovvero non cambia il valore della variabile nel workspace (passaggio per valore)

Funzioni

- **Esempio**
  - **function [y] = fattoriale1(n)**
    y = prod(1:n)
    endfunction

    Salvare la funzione nel file fattoriali.sci
  - **Carichiamo in Scilab il file**
  - **--> getf("fattoriali.sci") //obsoleto**
  - **--> exec("fattoriali.sci")**
  - **--> m = fattoriale1(5)**
    m = 120
  - **--> n1 = 2; n2 = 3; fattoriale1(n2)**
    ans = 6

Funzioni

- **E' possibile memorizzare e manipolare i valori restituiti da una funzione**
  - function [x1,x2] = risolvi_eq_2d(a,b,c)
    // calcola le radici di ax^2 + bx + c = 0
    delta = b^2 - 4*a*c
    x1 = (-b - sqrt(delta))/(2*a)
    x2 = (-b + sqrt(delta))/(2*a)
    endfunction
  - **Caricare in Scilab il file che contiene la funzione**

    dopo di che è possibile usare la funzione
  - **--> [r1,r2] = risolvi_eq_2d(1,2,1)**
    r2 = -1
    r1 = -1

Esercizio: Definire la funzione per il calcolo di

$$h(x) = \begin{cases} x + 2 & \text{se } x < -1 \\ x^2 + 3 & \text{se } -1 <= x < 6 \\ x^3 + 3*x^2 & \text{altrimenti} \end{cases}$$

Esercizio 1

Disegnare l'istogramma della distribuzione delle facce di un dado dopo 1000 lanci

Si definisce la funzione lancio

```
function y = lancio()
    y = int(rand() * 6 + 1);
endfunction
```

Si simula una serie di **k** lanci

```
function s = simula(k)
    for i=1:k
        s(i) = lancio()
    end
endfunction
```

Esercizio 1

Si Genera e si visualizza l'istogramma

```
function main()
      X = simula(1000);
      casi = linspace(1,6,7);
      histplot(casi,X, normalization=%f)
endfunction
```

1) Si memorizzino tutte le funzioni nel file dadi.sci
2) Caricare il file dadi.sci
3) Eseguire la funzione main()

Funzioni Ricorsive: Esercizio2...

Serie di Fibonacci

- 0 1 1 2 3 5 8 13 21 ....
- fib(0) = 0
- fib(1) = 1
- fib(n) = fib(n-1) + fib(n-2)

- Es. fib(2) = fib(1) + fib(0)=1

...Funzioni ricorsive: Esercizio2

```
// calcolo della serie di Fibonacci
// fib(0) = 0, fib(1) = 1,
// fib(n) = fib(n-1) + fib(n-2)
function f=fib(n)
  If n == 0 then
    f = 0
  elseif n == 1 then
    f=1
  else
    f=fib(n-1) + fib(n-2)
  end
endfunction
```

f(4) = f(3) + f(2) = f(2) + f(1) + 1 =
1 + 1 + 1 = 3

Funzioni ricorsive: Esercizio3

```
// calcolo del fattoriale in maniera ricorsiva

function f=fact(n)
  if n <= 1 then
    f=1
  else
    f=n*fact(n-1)
  end
endfunction
```

f(3) = 3 * f(2) = 3 * 2 * f(1) = 3 * 2 * 1 = 6