

Riconoscimento e Recupero dell'Informazione per Bioinformatica

Lab. 02 – Introduzione a Matlab (2)

Pietro Lovato

Corso di Laurea in Bioinformatica
Università di Verona - A.A. 2016/2017

Istruzione condizionale if

- Per eseguire un'istruzione soltanto se una certa espressione logica è vera
- Esempio: calcolo la radice quadrata di r solo se r è un numero non negativo:

```
>> if (r > 0)
>>     radice = sqrt(r);
>> else
>>     radice = NaN;
>> end
```

Istruzione condizionale if

- Operatori logici a disposizione

Operatore	Azione logica
&, &&	and
,	or
~	not
==	equal to
~=	not equal to
>=	greater or equal than
<=	lesser or equal than

Condizioni su vettori e matrici

- E' possibile effettuare operazioni logiche su vettori e/o matrici

```
>> x = [2 -1 8 0];
```

```
>> x > 0
```

```
ans =
```

```
1      0      1      0
```

Condizioni su vettori e matrici

- **Importante:** è possibile usare il risultato di un operazione logica su un vettore per accedere agli elementi il cui risultato è vero

```
>> x = [2 -1 8 0];
```

```
>> idx = x > 0;
```

```
>> x(idx) = 100
```

```
x =
```

```
100    -1    100     0
```

Condizioni su vettori e matrici

- Alternativa: comando `find` (per ottenere gli indici)

```
>> x = [2 -1; 8 0];
```

```
>> tmp = x > 0;
```

```
>> [riga, colonna] = find(tmp);
```

```
>> % riga = [1 2] colonna = [1 1]
```

Esercizio

- Capire il contenuto dello script

Lezione2Lab_es1_if.m e provare ad eseguire gli esercizi proposti.

Cicli di controllo `for`

- Per eseguire delle istruzioni in sequenza per ciascun valore $i = m, m + 1, \dots, n$ di una variabile i tra i limiti m e n

```
>> % Calcolo il prodotto scalare fra x e y
>> n = length(x);
>> ps = 0;
>> for i = 1:n
>>     ps = ps + x(i)*y(i);
>> end
```

Cicli di controllo `for`

- Varianti possibili: il ciclo `for` si può istanziare con un qualsiasi vettore

```
>> for i = 1:2:10
>>     fprintf('Valore di i: %i\n', i);
>> end
```

```
>> for i = [1 4 3 6 4 2]
>>     fprintf('Valore di i: %i\n', i);
>> end
```

Cicli di controllo `while`

- Per eseguire un'istruzione fintanto che una certa espressione logica è vera

```
>> ps = 0;  
>> i = 0; n = length(x);  
>> while (i < n)  
>>     i = i+1;  
>>     ps = ps + x(i)*y(i);  
>> end
```

Esercizio

- Capire il contenuto dello script

Lezione2Lab_es2_cicli.m e provare ad eseguire
gli esercizi proposti

Strutture dati avanzate

- Matlab permette di gestire matrici di dimensioni arbitrarie, non necessariamente 2D.

```
>> % Creo una matrice 3D:  
>> % è un "cubo" di lato 3  
>> A = zeros(3,3,3);  
>> A(:, :, 1) % Prima "fetta" del cubo
```

- Nota: ogni "fetta" ha la stessa dimensione

Strutture dati avanzate

- Celle: matrici i cui elementi sono matrici
- Ogni elemento di una cella può avere dimensioni diverse

```
>> A{1,1} = zeros(2,2);
```

```
>> A{2,1} = ones(3,3);
```

Strutture dati avanzate

- Per accedere al contenuto di una cella devo utilizzare le parentesi graffe

```
>> A{2,1} = [5 1; 2 7];
```

```
>> A{2,1}(1,1)
```

```
ans =
```

```
5
```

Funzioni

- Una funzione Matlab è una lista di comandi che necessita di **variabili di input** per essere eseguita e restituisce **variabili di output**.

Funzioni: linee guida

- Una funzione è contenuta in un file “.m” che ha lo stesso nome della funzione stessa
- Il file che contiene la funzione DEVE iniziare con: `function [output arguments] = nome_funzione(input arguments)`
- Tutte le variabili definite internamente sono locali (non verranno mostrate nel workspace)

Funzioni: esempio

- File “my_function.m”:

```
function f = my_function(x)
    f = x.^3 - 2*sin(x) + 1;
```

- Da prompt:

```
>> x = 0;
>> y = my_function(x)
ans =
     1
```

Esercizio

- Capire il contenuto dello script

Lezione2Lab_es3_strutture.m e provare ad eseguire
l'esercizio proposto