



Matlab: Strutture di Controllo

Informatica B

Tipo di dato logico

- ❑ È un tipo di dato che può avere solo due valori
 - ▶ true (vero) 1
 - ▶ false (falso) 0
- ❑ I valori di questo tipo possono essere generati
 - ▶ direttamente da due funzioni speciali (true e false)
 - ▶ dagli operatori relazionali
 - ▶ dagli operatori logici
- ❑ I valori logici occupano un solo byte di memoria (i numeri ne occupano 8)
- ❑ Esempio:
 - ▶ `a=true;`
 - ▶ `a` è un vettore 1x1 che occupa 1 byte e appartiene alla classe "tipo logico"

Operatori relazionali

- ❑ Gli operatori relazionali operano su tipi numerici o stringhe
- ❑ Forma generale: $a \text{ OP } b$
 - ▶ a, b possono essere espressioni aritmetiche, variabili, stringhe (della stessa dimensione)
 - ▶ OP: $==, \sim=, >, >=, <, <=$
- ❑ Esempi:
 - ▶ $3 < 4$ `true(1)`
 - ▶ $3 == 4$ `false(0)`
 - ▶ $'A' < 'B'$ `true(1)`
- ❑ Operatori relazionali possono essere usati per confrontare vettori con vettori della stessa dimensione o con scalari

Note

- ❑ Non bisogna confondere `==` e `=`
 - ▶ `==` è un operatore di confronto
 - ▶ `=` è un operatore di assegnamento
- ❑ La precisione finita può far commettere errori con `==` e `~`
`=`
 - ▶ `sin(0) == 0 -> 1`
 - ▶ `sin(pi) == 0 -> 0`
 - ▶ eppure logicamente sono vere entrambe!!
- ❑ Per i numeri piccoli conviene usare una soglia
 - ▶ `abs(sin(pi)) <= eps`

Vettori e stringhe

□ Esempi:

▶ $[1\ 0; -2\ 1] < 0$ $[false\ false; true\ false]$ ($[0\ 0; 1\ 0]$)

▶ $[1\ 0; -2\ 1] \geq [2\ -1; 0\ 0]$ $[false\ true; false\ true]$

□ Si possono confrontare stringhe di lunghezza uguale

▶ `'pippo' == 'pluto'` $[1\ 0\ 0\ 0\ 1]$

Operatori logici

- ❑ Forma generale: $a \text{ OP1 } b$ oppure $\text{OP2 } a$
 - ▶ a, b possono essere variabili, costanti, espressioni da valutare, scalari o vettori (dimensioni compatibili)
 - ▶ OP1: AND (&& o &), OR (|| o |), XOR (xor) e OP2: NOT (~)
- ❑ Se a e b sono numerici verranno interpretati come logici:
 - ▶ 0 come falso
 - ▶ tutti i numeri diversi da 0 come vero

a	b	a AND b	a OR b	NOT a	a XOR b
0	0	0	0	1	0
0	1	0	1	1	1
1	0	0	1	0	1
1	1	1	1	0	0

&& vs & e || vs |

- ❑ && (||) funziona con gli scalari e valuta prima l'operando più a sinistra. Se questo è sufficiente per decidere il valore di verità dell'espressione non va oltre
 - ▶ $a \ \&\& \ b$: se a è falso non valuta b
 - ▶ $a \ || \ b$: se a è vero non valuta b
- ❑ & (|) funziona con scalari e vettori e valuta **tutti** gli operandi prima di valutare l'espressione complessiva
- ❑ Esempio: $a/b > 10$
 - ▶ se b è 0 non voglio eseguire la divisione
 - ▶ $(b \neq 0) \ \&\& \ (a/b > 10)$ è la soluzione corretta: && controlla prima $b \neq 0$ e se questo è falso non valuta il secondo termine

Esempi

- ❑ “Hai tra 25 e 30 anni?”
 - ▶ $(eta \geq 25) \& (eta \leq 30)$
- ❑ Con i vettori:
 - ▶ $Voto = [12, 15, 8, 29, 23, 24, 27]$
 $C = (Voto > 22) \& (Voto < 25) \rightarrow C = [0 \ 0 \ 0 \ 0 \ 1 \ 1 \ 0]$
- ❑ Utile per contare quanti elementi soddisfano una condizione
 - ▶ $votiMedi = \text{sum}(Voto > 22 \& Voto < 25)$

Ordine tra gli operatori

- Un'espressione viene valutata nel seguente ordine:
 - ▶ operatori aritmetici
 - ▶ operatori relazionali da sinistra verso destra
 - ▶ NOT (\sim)
 - ▶ AND (& e $\&\&$) da sinistra verso destra
 - ▶ OR (| e $\|\|$) e XOR da sinistra verso destra

Funzioni logiche

Nome della funzione	Elemento restituito
<code>all(x)</code>	un vettore riga, con lo stesso numero di colonne dell'array <code>x</code> , che contiene 1, se la corrispondente colonna di <code>x</code> contiene tutti elementi non nulli, o 0 altrimenti
<code>any(x)</code>	un vettore riga, con lo stesso numero di colonne dell'array <code>x</code> , che contiene 1, se la corrispondente colonna di <code>x</code> contiene almeno un elemento non nullo, o 0
<code>isinf(x)</code>	un array delle stesse dimensioni di <code>x</code> con 1 dove gli elementi di <code>x</code> sono 'inf', 0 altrove
<code>isempty(x)</code>	1 se <code>x</code> è vuoto, 0 altrimenti
<code>isnan(x)</code>	un array delle stesse dimensioni di <code>x</code> con 1 dove gli elementi di <code>x</code> sono 'NaN', 0 altrove
<code>finite(x)</code>	un array delle stesse dimensioni di <code>x</code> , con 1 dove gli elementi di <code>x</code> sono finiti, 0 altrove
<code>ischar(x)</code>	1 se <code>x</code> è di tipo char, 0 altrimenti
<code>isnumeric(x)</code>	1 se <code>x</code> è di tipo double, 0 altrimenti
<code>isreal(x)</code>	1 se <code>x</code> ha solo elementi con parte immaginaria nulla, 0 altrimenti

Il costrutto if

```
if espressione1  
  istruzione 1-1  
  istruzione 1-2  
  .....  
elseif espressione2  
  istruzione 2-1  
  istruzione 2-2  
  .....  
.....  
else  
  istruzione k-1  
  istruzione k-2  
  .....  
end
```

I rami elseif e else non sono obbligatori!

Le istruzioni 1-1 e 1-2 vengono eseguite solo se vale espressione 1

Le istruzioni 2-1 e 2-2 vengono eseguite solo se vale espressione 2

Le istruzioni k-1 e k-2 vengono eseguite solo se non vale nessuna delle espressioni sopra indicate

Il costrutto switch

- ❑ L'istruzione condizionale switch consente una scrittura alternativa ad if/elseif/else
- ❑ Qualunque struttura switch può essere tradotta in un if/elseif/else equivalente

```
switch variabile (scalare o stringa)  
  case valore1  
    istruzioni caso 1  
  case valore2  
    istruzioni caso 2  
  ...  
  otherwise  
    istruzioni per i restanti casi  
end
```

Operatori relazionali per selezionare

- ❑ Gli operatori relazionali possono essere usati direttamente per selezionare gli elementi di un vettore
- ❑ Per esempio

$$x = [6, 3, 9] \text{ e } y = [14, 2, 9]$$

$$z = x(x < y)$$

- ▶ troviamo tutti gli elementi di x che sono minori del corrispondente elemento in y
- ▶ il risultato sarà $z = 6$.

Vettori logici per selezionare

- ❑ Quando utilizziamo un vettore logico come indice per un array, vengono estratti gli elementi corrispondenti ai valori 1 del vettore logico:
 - ▶ Se digitiamo $A(j)$, dove j è un vettore logico della stessa dimensione di A , otteniamo i valori di A corrispondenti agli indici degli 1 del vettore j
- ❑ Per creare un vettore logico NON basta creare un vettore di 0 e 1 (numeri), bisogna convertirlo con la funzione `logical`

```
i = [1,0,0,0,1];
```

```
j = logical(i)
```

```
A = [1 2 3 4 5];
```

```
A(j) → [1 5]
```

```
A(i) → errore
```

Altre funzioni logiche

❑ `i = find(x)` restituisce gli indici degli elementi non nulli dell'array `x`. `x` può essere un'espressione logica.

❑ Esempio

```
a = [ 5 6 7 2 10 ]
```

```
find(a>5) -> ans = 2 3 5
```

❑ Notate che `find` restituisce gli indici e non i valori degli array mentre usando i vettori logici come indici si ottengono i valori

❑ Esempio

```
x = [5, -3, 0, 0, 8];
```

```
y = [2, 4, 0, 5, 7];
```

```
values = y(x&y) -> values = [2 4 7]
```

```
indexes = find(x&y) -> values = [1 2 5]
```

Il ciclo while

while espressione

istruzioni da ripetere finché espressione è vera

end

- ❑ espressione deve essere inizializzata (avere un valore) prima dell'inizio del ciclo
- ❑ Il valore di espressione deve cambiare nelle ripetizioni
- ❑ Esempio: Calcoliamo gli interessi fino al raddoppio del capitale

```
value = 1000;
```

```
year = 0;
```

```
while value < 2000
```

```
    value = value * 1.08
```

```
    year = year + 1;
```

```
    fprintf('%g years: $%g\n', year, value)
```

```
end
```


Il ciclo for

```
for indice = inizio:incremento:fine  
    istruzioni  
end
```

- ❑ Se l'incremento viene omesso, viene usato 1 come valore di default

- ❑ Esempio – leggi 7 cifre e mettile in un vettore:

```
for digit = 1:7  
    number(digit) = input('enter value ');  
end
```

- ❑ Esempio - conto alla rovescia in secondi

```
time = input('how long? ');  
for count = time:-1:1  
    pause(1)  
    fprintf('%g seconds left \n',count)  
end  
disp('done')
```

Il ciclo for (2)

- ❑ È anche possibile usare un array per definire i valori dell'indice.
- ❑ Esempio – ciclo su una stringa:

```
for x = 'EGR106'  
    disp(x)  
end
```

Stampa E G R 1 0 6

- ❑ Esempio - ciclo per una matrice

```
board = [ 1 1 0 ; 1 1 -1 ; 0 1 0 ]  
for x = board  
    x  
end
```

X vale all'iterazione
iesima board(:,i)

Break e Continue

- ❑ I cicli contengono una serie di istruzioni che vogliamo ripetere
- ❑ Però potremmo aver bisogno di:
 - ▶ Saltare all'iterazione successiva
 - ▶ Terminare il ciclo
- ❑ Continue salta all'iterazione successiva
- ❑ Break interrompe l'esecuzione del ciclo

Esempio

- ❑ Acquisiamo numeri da tastiera finché non viene inserito un numero negativo. In ogni caso non accettiamo più di mille numeri:

```
vector = [ ]; %crea il vettore vuoto
for count = 1:1000 %Raccoglierà al max 1000 valori
    value = input('next number ');
    if value < 0
        break %Se value negativo usciamo dal ciclo
    else
        vector(count) = value;
    end
end
vector %visualizza il contenuto di vector
```