Matlab:
Logicals e Strutture di Controllo

Informatica B AA 2017/2018

Giacomo Boracchi

giacomo.boracchi@polimi.it

8 Novembre 2017

Tipo di Dato Logico

...e operazioni su vettori

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)

Esempi

```
>> a = true;
>> whos a
 Name   Size    Bytes  Class     Attributes
  a     1x1       1    logical
```

a è un vettore 1x1 che occupa 1 byte e appartiene alla classe "tipo logico"

```
>> a = 1>7
a =
     0
```

Operatori Relazionali

Operano su tipi numerici o stringhe.

Possono essere usati per confrontare

- due scalari
- due vettori aventi la stessa dimensione

Forma generale: a OP b

- a,b possono essere espressioni aritmetiche, variabili, stringhe (della stessa dimensione)
- OP: `==, ~=, >, >=, <, <=`

Esempi:

- `3 < 4` true(1)
- `3 == 4` false(0)
- `'A' < 'B'` true(1)

Note

Come in C: non confondere == e =

- == è un operatore di confronto
- = è un operatore di assegnamento

La precisione finita può produrre 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

Gli operatori relazionali tra vettori vengono applicati in maniera **puntuale**

Il risultato di un confronto tra v1 e v2 è un vettore v3 di tipo boolean, aventi le stesse dimensioni di v1 (e v2)

$$\text{v3 = (v1 >= v2);} \quad \text{v3}(i) = \begin{cases} 1, & \text{se v1(i)} \geq \text{v2(i)} \\ 0, & \text{se v1(i)} < \text{v2(i)} \end{cases}$$

Esempi:

```
>> [1 0; -2 1] < 0    [false false; true false] ([0 0; 1 0])
>> [1 0; -2 1] >= [2 -1; 0 0]      [false true; false true]
```

Si possono confrontare stringhe di lunghezza uguale

```
>> 'pippo'=='pluto'
   ans = [1 0 0 0 1]
```

Operatori Logici: Forma Generale

Operatori binari: **AND** (**&&**, oppure **&**, oppure **and**), **OR** (**||**, oppure **|**, oppure **or**), **XOR** (**xor**):

a OP1 b per la notazione simbolica
OP(a,b) per la notazione testuale

Operatori unari: **NOT** (**~**):

OP2 a

a,b possono essere variabili, costanti, espressioni da valutare, scalari o vettori (dimensioni compatibili)

Valori numerici di **a**, **b** vengono interpretati come logici:

- o come falso
- tutti i numeri diversi da o come vero

Richiamo, Tabelle di Verità

a	b	a && b	a || b	~a	`xor(a, 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

Or esclusivo: vero quando è vera solo uno delle due espressioni coinvolte
a XOR b == a OR b AND (~(a AND b))

&& 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 o non voglio eseguire la divisione
- **`(b~=0)&&(a/b>10)`** è la soluzione corretta: **`&&`** controlla prima **`b~=0`** e se questo è falso non valuta il secondo termine. Invece **`(b~=0)&(a/b>10)`** porterebbe ad una divisione per o quando **`b == 0`**

Esempi

"Hai tra 25 e 30 anni?"

```
(eta >= 25) & (eta <= 30)
```

Con i vettori:

```
Voto = [12, 15, 8, 29, 23, 24, 27]
 C = (Voto > 22) & (Voto < 25)
-> C = [0 0 0 0 1 1 0]
 D = (mod(Voto,2) == 0) | (Voto > 18)
-> D = [1 0 1 1 1 1 1]
 E = xor((mod(Voto,2)==0), (Voto>18))
-> E = [1 0 1 1 1 0 1]
```

Utile per contare quanti elementi soddisfano una condizione

```
nVoti= sum (Voto > 22 & Voto < 25)
```

Precedenze tra gli Operatori

Ogni espressione logica viene valutata rispettando il seguente ordine:

- operatori aritmetici
- operatori relazionali da sinistra verso destra
- NOT (`~`)
- AND (`&` e `&&`) da sinistra verso destra
- OR (`|` e `||`) e XOR da sinistra verso destra

Vettori Logici per Selezionare

I vettori logici possono essere usati per selezionare gli elementi di un array al posto di un vettore di indici

`nomeVettore(vettoreLogico)`

vengono estratti gli elementi di **`nomeVettore`** alle posizioni per cui **`vettoreLogico`** vale 1

Per esempio

```
>> x = [6,3,9]; y = [14,2,9];
>> b = x <= y ; % b = 1     0     1
>> z = x(b)
z =
     6     9
```

Esempi

Inizializzare a con i numeri da -10 a 20 con passo 3

Visualizzare solamente i numeri maggiori di 10

Portare a zero tutti gli elementi negativi

Sommare 10 ai numeri minori di 10

Cambiare il segno a tutte le occorrenze di -7 o 17

Note

`nomeVettore` e **`vettoreLogico`** devono avere la **stessa dimensione**

Per **creare un vettore logico non basta** creare un **vettore di 0 e 1** (numeri), bisogna convertirlo con la funzione logical

```
>> ii = [1,0,0,0,1];
>> jj = (ii == 1); %oppure jj = logical(ii)
>> A = [1 2 3 4 5];
>> A(jj) → [1 5]
>> A(ii) → Subscript indices must either
     be real positive integers or logicals.
```

Strutture di Controllo

Costrutto Condizionale: `if`, la sintassi

Il costrutto condizionale permette di eseguire istruzioni a seconda del valore di un'espressione booleana

`if, else, elseif, end` keywords

`expression` espressione booleana (vale 0 o 1)

`statement` sequenza di istruzioni da eseguire.

NB: il corpo è delimitato da **`end`**

NB: indentatura irrilevante

```
if(expression)
      statement
end
```

```
if(expression1)
      statement1
elseif(expression2)
      statement2
else
      statement0
end
```

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
```

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 non vale espressione1 ma vale espressione2

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

rami elseif e else non sono obbligatori!

Il Costrutto if

espressione1 può coinvolgere vettori:

- in tal caso **espressione1** è vera solo se tutti gli elementi di **espressione1** sono non nulli

Esempio

```
v = input('inserire vettore: ');
if (v >= 0)
    disp([num2str(v),' tutti pos. o nulli'])
elseif(v<0)
    disp([num2str(v),' tutti negativi']);
else
    disp([num2str(v),' sia pos. che neg.']);
end
```

Il Costrutto if

espressione1 può coinvolgere vettori:

- in tal caso **espressione1** è vera solo se tutti gli elementi di **espressione1** sono non nulli

Esempio

```
v = input('inserire vettore:');
if (v >= 0)
    disp([num2str(v),' tutti pos. o nulli'])
elseif(v<0)
    disp([num2str(v),' tutti negativi']);
else
    disp([num2str(v),' sia pos. che ne']);
end
```

Occorre inserire il vettore tra parentesi quadre.

Assumiamo un vettore riga

Esercizio

Scrivere un programma che richiede all'utente di inserire un numero e determina se corrisponde ad un anno bisestile

È possibile usare la funzione **`mod(a,b)`** che restituisce il resto della divisione tra **`a`** e **`b`**

Un anno è bisestile se è multiplo di 4 ma non di 100 oppure se è multiplo di 400

Esercizio

Scrivere un programma che richiede all'utente di inserire una stringa e controlla se questa è palindroma

Esercizio

Scrivere un programma che richiede all'utente di inserire una stringa e controlla se questa è palindroma

```
parola = input('inserire parola ' , 's');
str = [parola]
if (parola == parola(end : -1 : 1))
else
      str = [str , ' NON'])
end
str = [str , ' è palindroma'])
```

Non basta negare la condizione nell'if (come con gli scalari) perché il controllo ora è "se tutti gli elementi del vettore sono 1"

Il Costrutto switch

```
switch variabile %scalare o stringa
  case valore1
   istruzioni caso1
  case valore2
   istruzioni caso2
  ...
  otherwise
   istruzioni per i restanti casi
end
```

L'istruzione condizionale switch consente una scrittura alternativa ad if/elseif/else

Qualunque struttura switch può essere tradotta in un if/elseif/else equivalente

Note

Come per il C:

- **`valore1`** etc… devono essere delle espressioni costanti e si confrontano con **`variabile`** per verificarne l'uguaglianza

A differenza del C:

- solamente un caso viene eseguito: quando **`variabile`** corrisponde ad uno specifico **`valore`** non si eseguono tutti gli statement in cascata, si esce dal ciclo (è come se ci fosse sempre un break)
- è inutile usare break
- è possibile confrontare vettori
  - Sebbene **`variabile`** venga confrontata con **`valore1`** non è richiesto che queste abbiano la stessa lunghezza
  - Il case viene eseguito se tutti gli elementi corrispondono

Note

È possibile mettere più valori nel case, separati da graffe

```
str = 'pluto';
switch str
    case {'pippo', 'pluto', 'paperino',
'clarabella'}
        disp('Walt Disney')
    otherwise
        disp('no Walt Disney')
end
```

In questo caso basta che ci sia un match tra str e un elemento tra le parentesi graffe

Il Ciclo while

```
while expr
  istruzioni da ripetere finché expr è
  vera
end
```

expr assume valore 0 o 1e può contenere con operatori relazionali (==, <, >, <=, >=, ~=)

expr deve essere inizializzata (avere un valore) prima dell'inizio del ciclo

Quando **expr** coinvoge vettori si ha che **expr** è vera se tutti gli elementi sono non nulli (come per if)

Il valore di espressione deve cambiare nelle ripetizioni

Esempio

Stampare, utilizzando un ciclo i numeri da 100 a 1

```
n = 100;
while(n > 0)
    disp(n);
    n = n - 1;
end
```

In alternativa

```
[100 : - 1 : 1]'
```

Esempio

Calcoliamo gli interessi fino al raddoppio del capitale, si assuma un interesse annuo del 8%

Esempio

% il quadrato di N è uguale alla somma dei primi N numeri dispari, calcolare il quadrato di un nr inserito da utente (<100)

Esempio

Richiedere all'utente di inserire un numero e, se questo corrisponde ad un anno bisestile, chiederne un altro. Il programma termina quando viene inserito un numero che non corrisponde ad un anno bisestile.
Al termine, il programma scrive quanti anni bisestili ha inserito l'utente

Il ciclo for

```
for variabile = array
    istruzioni
end
```

Tipicamente **`array`** è un vettore, quindi **`variabile`** assume valori scalari

- Alla prima iterazione **`variabile`** è **`array(1)`**
- Alla seconda iterazione **`variabile`** è a **`array(2)`**
- All'ultima iterazione **`variabile`** è **`array(end)`**

NB: Non esiste alcuna condizione da valutare per definire la permanenza nel ciclo. Il numero di iterazioni dipende dalle dimensioni di array

NB: se **`array`** è un'espressione booleana viene scandito come il vettore logico.

Il ciclo for

Non è equivalente al **while**, ha meno potere espressivo: ad esempio non è possibile eseguire infinite volte il corpo di un **for**

Ogni **for** può essere scritto come un **while**

```
for c = 'ciao'
    disp(c)
end
```

c assumerà ad ogni iterazione un carattere diverso nel vettore 'ciao'

Il ciclo for

Non è equivalente al **while**, ha meno potere espressivo: ad esempio non è possibile eseguire infinite volte il corpo di un **for**

Ogni **for** può essere scritto come un **while**

```
for c = 'ciao'
    disp(c)
end
```

```
vet = 'ciao'
ii = 1;
while (ii <=length(vet))
   disp(vet(ii))
   ii = ii + 1;
end
```

Il ciclo for

Non è equivalente al **while**, ha meno potere espressivo: ad esempio non è possibile eseguire infinite volte il corpo di un **for**

Ogni **for** può essere scritto come un **while**

```
for c = 'ciao'
    disp(c)
end
```

```
vet = 'ciao'
ii = 1;
while (ii <=length(vet))
    disp(vet(ii))
    ii = ii + 1;
end
```

Occorre usare un indice esplicito ii

Occorre scorrere il vettore calcolandone la lunghezza

Occorre incrementare ii

Il ciclo for

Non è equivalente al **while**, ha meno potere espressivo: ad esempio non è possibile eseguire infinite volte il corpo di un **for**

Ogni **for** può essere scritto come un **while**

```
for c = 'ciao'
      disp(c)
end
```

```
vet = 'ciao'
ii = 1;
while (ii <= length(vet))
      disp(vet(ii))
      ii = ii + 1;
end
```

Per scorrere un vettore noto, il ciclo for è molto più comodo del while, se invece il numero di iterazioni da eseguire non è noto a priori è preferibile usare while

Il ciclo for , la variabile del ciclo

```
for variabile = array
    istruzioni
end
```

array può essere generato "al volo", **molto spesso è un vettore riga** definito tramite l'operatore di incremento regolare, i.e., "inizio : step : fine"

Esempi

```
% leggi 7 numeri e mettili in un vettore:
```

```
% stampa conto alla rovescia in secondi
```

Esempi

```
% leggi 7 numeri e mettili in un vettore:
   for n = 1:7
      number(n) = input('enter value ');
   end

% stampa 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 , la variabile del ciclo (2)

```
for variabile = array
    istruzioni
end
```

Quando **`array`** è una matrice, il ciclo viene eseguito un numero volte pari al numero di colonne di **`array`** e ogni volta **`variabile`** assume il valore di una colonna

- Alla prima iterazione è **`variabile`** è **`array(:, 1)`**
- Alla seconda iterazione è **`variabile`** è **`array(:, 2)`**
- All'ultima iterazione è **`variabile`** è **`array(:, end)`**

N.B. Quando **`array`** è un vettore colonna, questo viene considerato una matrice e si esegue una sola iterazione in cui **`variabile`** è uguale ad **`array`**

Il ciclo for , la variabile del ciclo

Esempio di for su una **una matrice**

```
board = [ 1 1 1 ; 1 1 -1 ; 0 1 0 ];
for x = board
    disp('colonna:')
    x %stampa in ogni iterazione una colonna di board
end
```

```
colonna:

x =

     1

     1

     0

colonna:
x =

     1

     1

     1

colonna:

x =

     1

    -1

     0
```

Esercizio

Si assuma che il vettore voti contenga i risultati della prima prova in itinere (una traccia del primo A)

- Si calcoli la media
- Si calcoli la media dei voti sufficienti (>= 8)
- Si calcoli la varianza dei voti sufficienti (si ricorda che la varianza si ottiene come la media degli scarti quadratici dalla media)
- Si calcoli il numero di voti maggiori o uguali a 15

Ciclo for per accodare vettori

```
% leggi 7 numeri e mettili in un vettore:
   for n = 1:7
      number(n) = input('enter value ');
   end
```

Questa soluzione non permette all'utente di inserire un

```
>> enter value [1 13]
```

```
In an assignment  A(I) = B, the number of
elements in B and I must be the same
```

Per risolvere questo problema si può acquisire l'input in una variabile temporanea **`temp`** (non un elemento di un vettore) e poi accodare il contenuto di **`temp`** in una variabile accumulatore

```
% leggi 7 numeri e mettili in un vettore
clear
vettore = [];
for ii = [1 : 7]
    temp = input('inserire numero ');
    vettore = [vettore, temp];
end
disp(vettore)
```

```
% leggi 7 numeri e mettili in un vettore
clear
vettore = [];
for ii = [1 : 7]
    temp = input('inserire numero ');
    vettore = [vettore, temp];
end
disp(vettore)
```

È necessario inizializzare **vettore** a vuoto altrimenti la prima esecuzione dell'istruzione **vettore = [temp, vettore];** genera un errore perché **vettore** non esiste

```
% leggi 7 numeri e mettili in un vettore
clear
vettore = [];
for ii = [1 : 7]
    temp = input('inserire numero ');
    vettore = [vettore, temp];
end
disp(vettore)
```

In questo modo il numero inserito viene accodato a **vettore**

```
% leggi 7 numeri e mettili in un vettore
clear
vettore = [];
for ii = [1 1 1 1 1 1 1]
    temp = input('inserire numero ');
    vettore = [vettore, temp];
end
disp(vettore)
```

Non si usa più il valore dell'indice **ii** nel ciclo, quindi basta mettere un vettore di lunghezza 7 per garantire che il ciclo venga eseguito 7 volte

```
% leggi 7 numeri e mettili in un vettore
clear
vettore = [];
for ii = [1 : 7]
    temp = input('inserire numero ');
    vettore = [temp, vettore];
end
disp(vettore)
```

In questo modo il numero inserito messo prima di **vettore** e quindi i numeri inseriti vengono stampati in ordine inverso

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

Come nel C:

- 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:

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
```

Esercizio

Scrivere un programma che richiede in ingresso un intero N e restituisce l'N-simo numero della sequenza di Fibonacci (definita come segue)

- $F(0) = 1$
- $F(1) = 1$
- $F(n) = F(n-1) + F(n-2)$

Richiedere un secondo numero M e dire se è uno dei primi N numeri di Fibonacci ed, in caso negativo, restituire i due numeri più vicini.