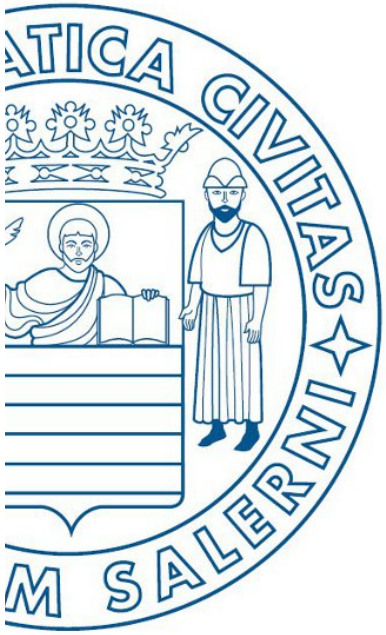


UNIVERSITÀ DEGLI STUDI DI SALERNO

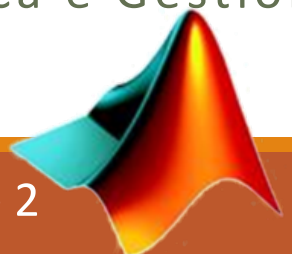
Fondamenti di Informatica

Introduzione alla programmazione in MATLAB: Parte 2
(Strutture di controllo selettive ed iterative)

Prof. Christian Esposito

Corso di Laurea in Ingegneria Meccanica e Gestionale (Classe I)

A.A. 2017/18



MATLAB

OUTLINE

- Operatori Relazionali e Logici
- Strutture di controllo del flusso
  - Costrutto IF
  - Costrutto SWITCH-CASE
- Strutture iterative in MATLAB
  - Ciclo FOR
  - Ciclo WHILE
- Istruzioni Break e continue

Operatori Relazionali – 1/4

- Basati sui principi della Logica Booleana
- Il loro scopo è quello di fornire risposte di tipo vero/falso a domande riguardanti il confronto di relazioni tra oggetti
- Più precisamente, gli *operatori relazionali* **confrontano** relazioni tra
  - Tipi numerici
    - Array, matrici e scalari
  - Stringhe
  - Intere espressioni

- Il risultato di una comparazione può essere **vero** (1) oppure **falso** (0)
- In MATLAB gli operatori relazionali possono operare direttamente anche su array e matrici

Operatori Relazionali – 2/4

- In MATLAB
  - Un **confronto falso** → da risultato **0**
  - Un **confronto vero** → da risultato **1** (o un valore *diverso da 0*)

Operatore	Descrizione
>	Maggiore di
<	Minore di
>=	Maggiore o uguale di
<=	Minore o uguale di
==	Uguale a
~=	Diverso da

Operatori Relazionali – 2/4

- In MATLAB
  - Un **confronto falso** → da risultato **0**
  - Un **confronto vero** → da risultato **1** (o un valore *diverso da 0*)

Operatore	Descrizione
>	Maggiore di
<	Minore di
>=	Maggiore o uguale di
<=	Minore o uguale di
==	**Uguale a**
~=	Diverso da

NOTA IMPORTANTE:
L'operatore relazionale «**Uguale a**» è composto da **due simboli uguale: ==**, da non confondere con l'operatore di assegnazione con un uguale **=**

Operatori Relazionali – 2/4

- In MATLAB
  - Un **confronto falso** → da risultato **0**
  - Un **confronto vero** → da risultato **1** (o un valore *diverso da 0*)

Operatore	Descrizione
>	Maggiore di
<	Minore di
>=	**Maggiore o uguale di**
<=	**Minore o uguale di**
==	Uguale a
~=	Diverso da

NOTA IMPORTANTE:
Usare >=
Non usare =>

Usare <=
Non usare =<

Operatori Relazionali – 3/4

- ***Esempio 1 (== applicato a scalari)***

```
>> a=5
a =
     5
>> b=9
b =
     9
>> a==b
ans =
     0
```

Operatori Relazionali – 3/4

- ***Esempio 2***

```
>> x = 5;
>> y = 6;
```

Espressione	Valore
x < y	1
x <= y	1
4 > 5	0
x == 5	1
x ~= y	1
x + 1 – y == 0	1

Operatori Relazionali – 4/4

- ***Esempio 3 (== applicato a matrici)***

```
>> M1=[4 3 3; 2 8 1; -1 85 28]
M1 =
     4     3     3
     2     8     1
    -1    85    28
>> M2=[2 3 4; 3 3 1; 5 6 28]
M2 =
     2     3     4
     3     3     1
     5     6    28
>> M1==M2
ans =
     0     1     0
     0     0     1
     0     0     1
```

Operatori Relazionali – 4/4

- ***Esempio 4 (> applicato ad array)***

```
>> x = 1 : 8
x =
   1    2    3    4    5    6    7    8

>> y = 8 : -1 : 1

y =
   8    7    6    5    4    3    2    1

>> bool_res = x > y

bool_res =
     0    0    0    0    1    1    1    1
```

Operatori Logici

- Gli operatori logici forniscono un modo per combinare o negare espressioni relazionali
  - Operatori logici di base: AND, OR e NOT

Operatore	Descrizione	Note
&&	**AND**	**A && B da risultato 1 se A e B sono *entrambi* uguali a 1; A && B da risultato 0, altrimenti**
||	**OR**	**A || B da risultato 1 se *almeno uno* tra A e B è uguale a 1; A || B da risultato 0, altrimenti**
~	**NOT**	**~A da risultato 0, se A è uguale a 1 ~A da risultato 1, se A è uguale a 0**

- Come visto nella prima parte del corso, ciascun operatore logico può essere specificato mediante una tavola di verità

Ordine di Precedenza dei Tipi di Operatori

Livello di precedenza	Tipo di operatore
Primo	Parentesi tonde; sono valutate a partire dalla coppia di parentesi più interne
Secondo	Operatori aritmetici e operatore NOT (~); sono valutati da sinistra a destra
Terzo	Operatori relazionali; sono valutati da sinistra a destra
Quarto	Operatore logico AND
Quinto	Operatore logico OR

Operatori Logici: Esempi – 1/2

- Espressione booleana che è vera solo se **x** è maggiore di 10 AND minore di 20

```
>> x = 5;
>> (10 < x) && (x < 20)

ans =

     0
```

Operatori Logici: Esempi – 2/2

- È possibile controllare la precedenza usando le parentesi

```
>> x = false;
>> y = true;

>>  x && y || true

        0     OR    1

ans =
     1


>> x && (y || true)

    0       OR    1

ans =
     0
```

Operatori Logici
(Elemento per Elemento)

- Gli operatori logici possono essere applicati agli array

Operatore logico	Descrizione
&	**AND elemento per elemento**
|	**OR elemento per elemento**
~	**NOT elemento per elemento**

- N.B.
  - **Non va confuso** l'operatore & con l'operatore &&
  - **Non va confuso** l'operatore | con l'operatore ||

Operatori Logici Elemento per Elemento (Esempio)

```
>> x = 1 : 8
x =
   1   2   3   4   5   6   7   8

>> y = 8 : -1 : 1
y =
   8   7   6   5   4   3   2   1

>> bool_res = x > y
bool_res =
   0   0   0   0   1   1   1   1

>> bool_res_neg = ~(x > y)
bool_res_neg =
   1   1   1   1   0   0   0   0

>> (x > y) & (y <= 2)
ans =
   0   0   0   0   0   0   1   1

>> (x < 2) | (y < 2)
ans =
   1   0   0   0   0   0   0   1
```

Operatori Logici
(Indicizzazione di Array)

Ottenere la matricola di tutti gli operai che guadagnano più di 5000 euro e lavorano meno di 10 giorni al mese

```
>> salario = [1000, 7000, 6000, 3000, 8000];
>> giorni_lavorativi = [20, 8, 28, 5, 7];
>> matricola_operai = [10001, 10002, 10003, 10004, 10005];

>> salario > 5000
ans =
     0   1   1   0   1

>> giorni_lavorativi < 10
ans =
     0   1   0   1   1

>> (salario > 5000) & (giorni_lavorativi < 10)
ans =
     0   1   0   0   1

>> matricola_operai ((salario > 5000) & (giorni_lavorativi < 10))

ans =
     10002   10005
```

Operatori Logici
(Indicizzazione di Matrici)

```
>> x = [8 2 3; 1 5 0]
x =
    8    2    3
    1    5    0

>> x >= 5
ans =
     1    0    0
     0    1    0

>> x(x >= 5)
ans =
     8
     5
```

Notare la differenza con il comando `find`

```
>> find(x >= 5)
ans =
     1
     4
```

Notare che nel risultato finale non ci sono differenze

```
>> x(find(x >= 5))
ans =
     8
     5
```

Strutture fondamentali di MATLAB (1)

- Analogamente ad altri linguaggi di programmazione, MATLAB dispone di tre strutture fondamentali

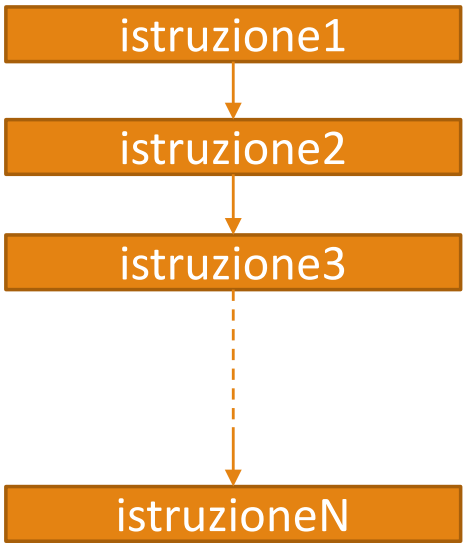
- Sequenza
- Selezione
- Iterazione

Strutture fondamentali di MATLAB (2)

- **Sequenza**
  - La sequenza è definita dalla sequenza lessicografica delle istruzioni

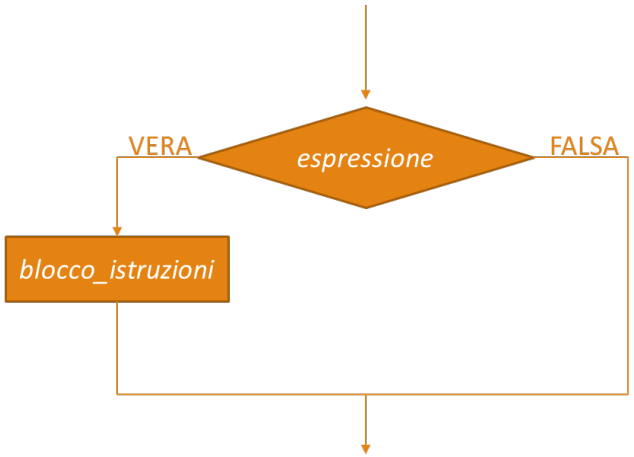
Esecuzione TOP-DOWN
(alto verso il basso)

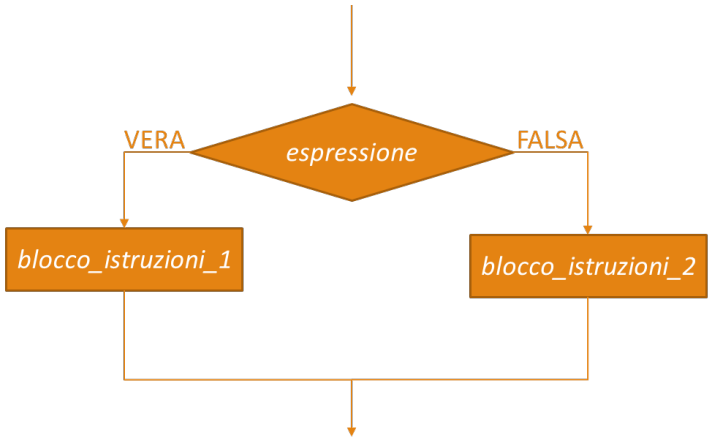
```
istruzione1
istruzione2
istruzione3
.
.
.
istruzioneN
```

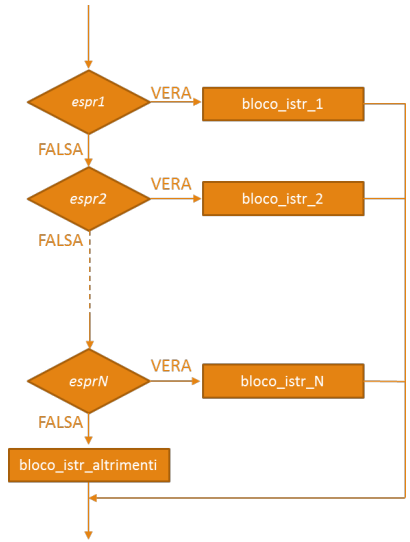


Strutture fondamentali di MATLAB (3)

- **Selezione**
  - In MATLAB, la strutture di selezione del flusso viene delineata dal costrutto IF

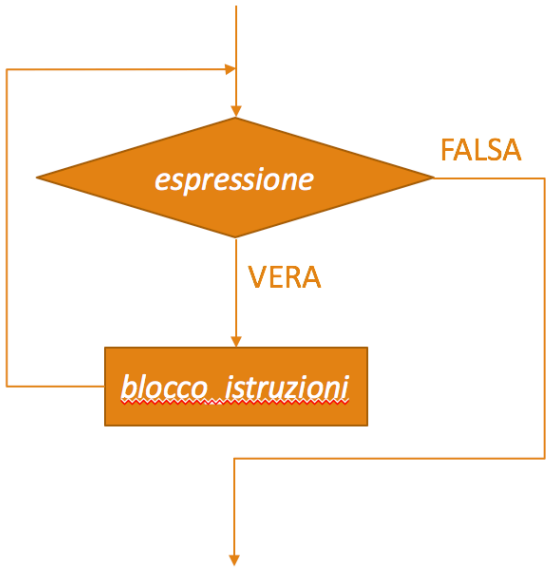


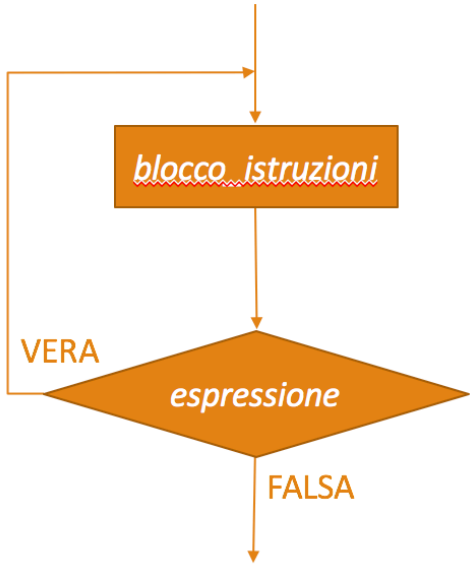




Strutture fondamentali di MATLAB (4)

- **Iterazione**
  - Le strutture iterative vengono delineate dai cicli FOR e WHILE
    - Verranno affrontate nella prossima lezione





Costrutto IF (1)

- **Selezione semplice**:

```
if espressione
    blocco_istruzioni
end
```

Costrutto IF (1)

- **Selezione semplice**:

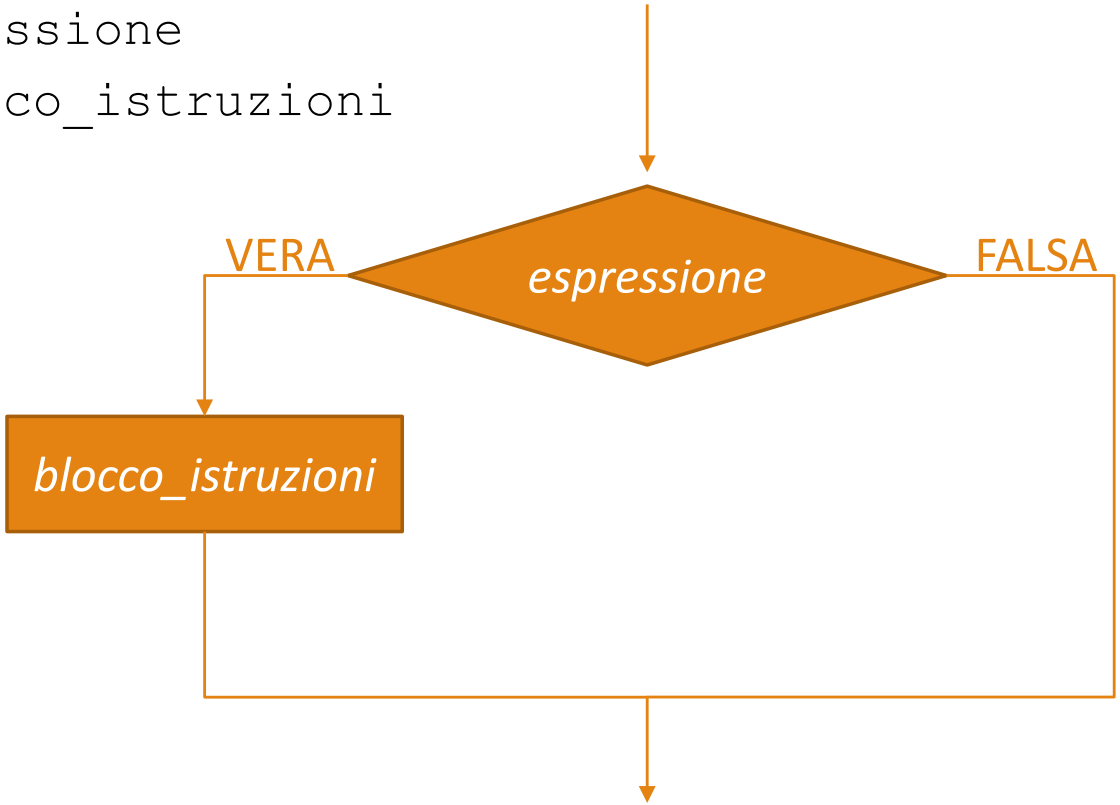
```
if espressione
    blocco_istruzioni
end
```

- **Le istruzioni** del blocco **sono eseguite** solo se l'espressione risulterà **vera**
  - Se risulterà **falsa**, le istruzioni **non saranno eseguite**

Costrutto IF (1)

- **Selezione semplice**:

```
if espressione
    blocco_istruzioni
end
```



Costrutto IF (2)

- **Selezione semplice**:

```
if espressione
    blocco_istruzioni
end
```

- ***Esempio***

```
x = input('Inserisci x: ');
y = input('Inserisci y: ');

if x == y
    disp('x e y sono uguali');
end
```

Costrutto IF (3)

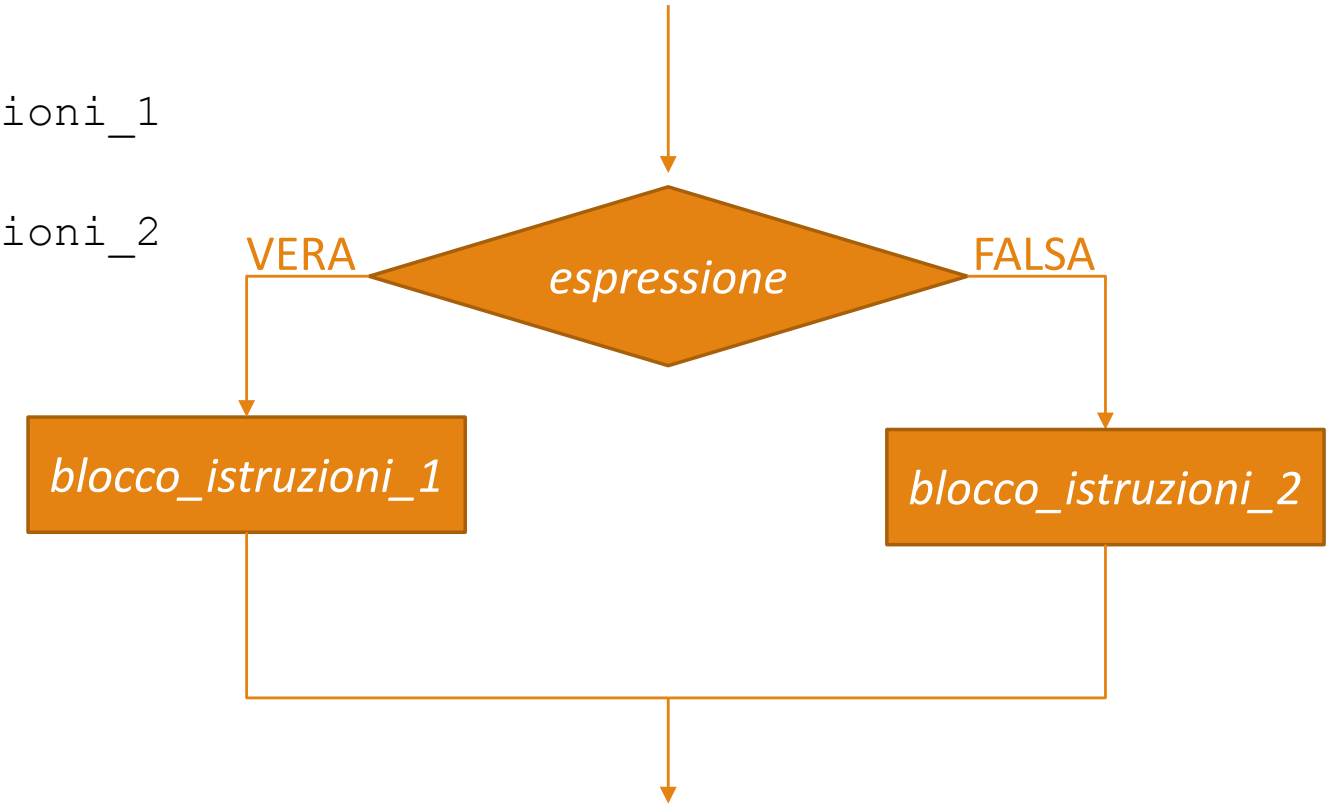
- **Selezione a due vie**:

```
if espressione
    blocco_istruzioni_1
else
    blocco_istruzioni_2
end
```

Costrutto IF (3)

- **Selezione a due vie**:

```
if espressione
    blocco_istruzioni_1
else
    blocco_istruzioni_2
end
```



Costrutto IF (3)

- **Selezione a due vie**:

```
if espressione
    blocco_istruzioni_1
else
    blocco_istruzioni_2
end
```

- ***Esempio 1***

```
x = input('Inserisci x: ');

if x > 0
    disp('Hai inserito un valore positivo');
else
    disp('Hai inserito un valore negativo');
end
```

Costrutto IF (4)

- **Selezione a due vie**:

```
if espressione
    blocco_istruzioni_1
else
    blocco_istruzioni_2
end
```

- ***Esempio 2***

```
n = input('Inserisci n: ');

if n < 100
    disp([num2str(n), ' è MINORE di 100']);
else
    disp([num2str(n), ' è MAGGIORE O UGUALE di 100']);
end
```

Costrutto IF (5)

- **Selezione a due vie**:

```
if espressione
    blocco_istruzioni_1
else
    blocco_istruzioni_2
end
```

- ***Esempio 3***

```
if voto_esame < 18
    disp('GRRRRRRRR...Il prof mi ha preso in antipatia!!!');
else
    disp('E vai!!! Ho superato l'esame superato! Bye bye prof');
end
```

Costrutto IF (6)

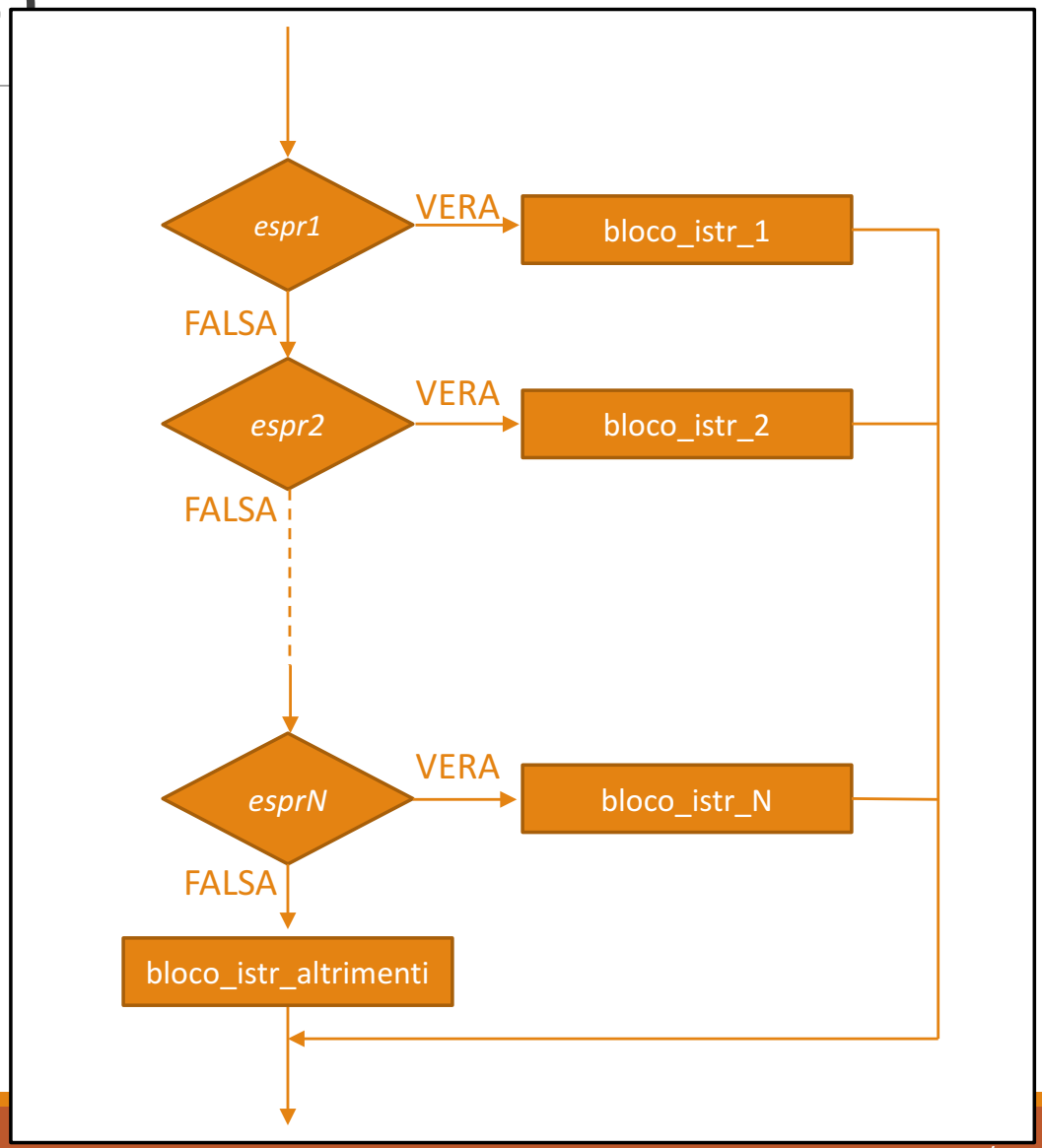
- **Selezione a «cascata»** :

```
if espressione1
    blocco_istruzioni_1
elseif espressione2
    blocco_istruzioni_2
.
.
.
elseif espressioneN
    blocco_istruzioni_N
else
    blocco_istruzione_altrimenti
end
```

Costrutto IF (6)

- **Selezione a «cascata»** :

```
if espressione1
    blocco_istruzioni_1
elseif espressione2
    blocco_istruzioni_2
.
.
.
elseif espressioneN
    blocco_istruzioni_N
else
    blocco_istruzione_altrimenti
end
```



Costrutto IF (7)

- **Selezione a «cascata» (Esempio)**

```
if prezzo >= 999
    perc_sconto = 35;
elseif prezzo >= 599 && prezzo < 999
    perc_sconto = 25;
elseif prezzo < 599
    perc_sconto = 19;
elseif prezzo <= 0.99
    perc_sconto = 0;
    disp('No sconto! Spendi di più per averlo!!')
else
    disp('prezzo non corretto!')
end
```

Costrutto IF (7)

- **Selezione a «cascata» (Esempio)**

```
if prezzo >= 999
    perc_sconto = 35;
elseif prezzo >= 599 && prezzo < 999
    perc_sconto = 25;
elseif prezzo < 599
    perc_sconto = 19;
elseif prezzo <= 0.9
    perc_sconto
    disp('No sc
else
    disp('prezz
end
```

NOTA: Equivale a
$599 \leq prezzo < 999$

Costrutto IF (8)

- ***Esempio* IF *annidati (Anno Bisestile)***

Le **condizioni** affinché un anno sia bisestile sono:

- L'anno deve essere **divisibile per 4** e **inoltre**
- nel caso in cui sia **divisibile per 100** deve essere ***anche*** **divisibile per 400**

Costrutto IF (8)

Le **condizioni** affinché un anno sia bisestile sono:

- L'anno deve essere **divisibile per 4** e **inoltre**
- nel caso in cui sia **divisibile per 100** deve essere ***anche*** **divisibile per 400**

◦ ***Esempio* IF *annidati***

Costrutto IF (8)

Le **condizioni** affinché un anno sia bisestile sono:

- L'anno deve essere **divisibile per 4** e **inoltre**[1]
- nel caso in cui sia **divisibile per 100** deve essere ***anche*** **divisibile per 400**

- Se l'anno *è divisibile per 4*
  - ***devo fare ulteriori verifiche***[1]**…**
- Altrimenti
  - L'anno ***NON E' BISESTILE***
- Fine (Se)

◦ ***Esempio* IF *annidati***

Costrutto IF (8)

Le **condizioni** affinché un anno sia bisestile sono:
- L'anno deve essere **divisibile per 4** e inoltre
- nel caso in cui sia **divisibile per 100** ***deve essere anche[2]*** **divisibile per 400**

◦ ***Esempio* IF *annidati***

- Se l'anno *è divisibile per 4*
  - Se l'anno *è divisibile per 100*
    - ***devo fare ulteriori verifiche[2]…***
  - Altrimenti
    - L'anno ***E' BISESTILE***
  - Fine (Se)
- Altrimenti
  - L'anno ***NON E' BISESTILE***
- Fine (Se)

Costrutto IF (8)

Le **condizioni** affinché un anno sia bisestile sono:

- L'anno deve essere **divisibile per 4** e inoltre
- nel caso in cui sia **divisibile per 100** deve essere anche **divisibile per 400**

◦ ***Esempio* IF *annidati***

- Se l'anno *è divisibile per 4*
  - Se l'anno *è divisibile per 100*
    - Se l'anno *è divisibile per 400*
      - L'anno ***E' BISESTILE***
    - Altrimenti
      - L'anno ***NON E' BISESTILE***
    - Fine (Se)
  - Altrimenti
    - L'anno ***E' BISESTILE***
  - Fine (Se)
- Altrimenti
  - L'anno ***NON E' BISESTILE***
- Fine (Se)

Costrutto IF (8)

Le **condizioni** affinché un anno sia bisestile sono:

- L'anno deve essere **divisibile per 4** e inoltre
- nel caso in cui sia **divisibile per 100** deve essere anche **divisibile per 400**

◦ ***Esempio* IF *annidati***

- Se l'anno *è divisibile per 4*
    - Se l'anno *è divisibile per 100*
        - Se l'anno *è divisibile per 400*
            - L'anno ***E' BISESTILE***
        - Altrimenti
            - L'anno ***NON E' BISESTILE***
        - Fine (Se)
    - Altrimenti
        - L'anno ***E' BISESTILE***
    - Fine (Se)
- Altrimenti
    - L'anno ***NON E' BISESTILE***
- Fine (Se)

Costrutto IF (8)

Le **condizioni** affinché un anno sia bisestile sono:
- L'anno deve essere **divisibile per 4** e inoltre
- nel caso in cui sia **divisibile per 100** deve essere anche **divisibile per 400**

- Se l'anno *è **divisibile** per 4*
  - Se l'anno *è divisibile per 100*
    - Se l'anno *è divisibile per 400*
      - L'anno ***E' BISESTILE***
    - Altrimenti
      - L'anno ***NON E' BISES***
    - Fine (Se)
  - Altrimenti
    - L'anno ***E' BISESTILE***
  - Fine (Se)
- Altrimenti
  - L'anno ***NON E' BISESTILE***
- Fine (Se)

Come si verifica se un numero **N** ***è divisibile*** per un numero **M**?

◦ ***Esempio* IF *annidati***

Costrutto IF (8)

Le **condizioni** affinché un anno sia bisestile sono:
- L'anno deve essere **divisibile per 4** e inoltre
- nel caso in cui sia **divisibile per 100** deve essere anche **divisibile per 400**

◦ ***Esempio* IF *annidati***

- Se l'anno *è* ***divisibile*** *per 4*
  - Se l'anno *è divisibile per 100*
    - Se l'anno *è divisibile per 400*
      - L'anno ***E' BISESTILE***
    - Altrimenti
      - L'anno ***NON E' BISES***
    - Fine (Se)
  - Altrimenti
    - L'anno ***E' BISESTILE***
  - Fine (Se)
- Altrimenti
  - L'anno ***NON E' BISESTILE***
- Fine (Se)

Come si verifica se un numero **N *è divisibile*** per un numero **M**?

Se **N *modulo* M è uguale a 0** allora **N è divisibile per M**

(*modulo* → resto della divisione tra N e M)

Costrutto IF (8)

Le **condizioni** affinché un anno sia bisestile sono:
- L'anno deve essere **divisibile per 4** e inoltre
- nel caso in cui sia **divisibile per 100** deve essere anche **divisibile per 400**

◦ ***Esempio* IF *annidati***

- Se l'anno *è* ***divisibile*** *per 4*
  - Se l'anno *è divisibile per 100*
    - Se l'anno *è divisibile per 400*
      - L'anno ***E' BISESTILE***
    - Altrimenti
      - L'anno ***NON E' BISES***
    - Fine (Se)
  - Altrimenti
    - L'anno ***E' BISESTILE***
  - Fine (Se)
- Altrimenti
  - L'anno ***NON E' BISESTILE***
- Fine (Se)

Come si verifica se un numero **N *è divisibile*** per un numero **M**?

Se **N *modulo* M è uguale a 0** allora **N è divisibile per M**

(*modulo* → resto della divisione tra N e M)

Modulo in MATLAB: `mod(N,M)`

Costrutto IF (8)

Le **condizioni** affinché un anno sia bisestile sono:

- L'anno deve essere **divisibile per 4** e inoltre
- nel caso in cui sia **divisibile per 100** deve essere anche **divisibile per 400**

- Se **mod(anno, 4) == 0** % è divisibile per 4?
  - Se **mod(anno, 100) == 0** % è divisibile per 100?
    - Se **mod(anno, 400) == 0** % è divisibile per 400?
      - L'anno ***E' BISESTILE***
    - Altrimenti
      - L'anno ***NON E' BISESTILE***
    - Fine (Se)
  - Altrimenti
    - L'anno ***E' BISESTILE***
  - Fine (Se)
- Altrimenti
  - L'anno ***NON E' BISESTILE***
- Fine (Se)

◦ ***Esempio IF annidati***

Costrutto IF (8)

Le **condizioni** affinché un anno sia bisestile sono:
- L'anno deve essere **divisibile per 4** e inoltre
- nel caso in cui sia **divisibile per 100** deve essere anche **divisibile per 400**

◦ ***Esempio IF annidati***

- Se **mod(anno, 4) == 0**
  - Se **mod(anno, 100) == 0**
    - Se **mod(anno, 400) == 0**
      - L'anno ***E' BISESTILE***
    - Altrimenti
      - L'anno ***NON E' BISESTILE***
    - Fine (Se)
  - Altrimenti
    - L'anno ***E' BISESTILE***
  - Fine (Se)
- Altrimenti
  - L'anno ***NON E' BISESTILE***
- Fine (Se)

◦ ***Codice MATLAB***

```
if mod(anno,4) == 0
    if mod(anno, 100) == 0
        if mod(anno, 400) == 0
            disp('Bisestile');
        else
            disp('Non bisestile');
        end
    else
        disp('Bisestile');
    end
else
    disp('Non bisestile');
end
```

Costrutto IF (8)

- ***Esempio IF annidati***

```
anno = input('Inserisci l''anno: '); % input

if mod(anno,4) == 0
    if mod(anno, 100) == 0
        if mod(anno, 400) == 0
            disp([num2str(anno), ' è bisestile']);
        else
            disp([num2str(anno), ' non è bisestile']);
        end
    else
        disp([num2str(anno), ' è bisestile']);
    end
else
    disp([num2str(anno), ' non è bisestile']);
end
```

Esempi d'uso

M-file Script: ***anno_bisestile.m***

```
>> anno_bisestile
Inserisci l'anno: 2000
2000 non è bisestile

>> anno_bisestile
Inserisci l'anno: 2004
2004 è bisestile

>> anno_bisestile
Inserisci l'anno: 2016
2016 è bisestile

>> anno_bisestile
Inserisci l'anno: 2018
2018 non è bisestile
```

Costrutto SWITCH-CASE (1)

- Il costrutto `switch-case` permette di scegliere di eseguire un gruppo di istruzioni in base al valore di una variabile

```
switch variabile
    case valore1
        blocco_istruzioni_valore_1
    case valore2
        blocco_istruzioni_valore_2
     .
     .
     .
    case valoreN
        blocco_istruzioni_valore_N
    otherwise
        blocco_istruzioni_otherwise
end
```

- Lo `switch` risulta essere utile per le enumerazioni e poco appropriato per gli intervalli

Costrutto SWITCH-CASE (2)

- ***Esempio 1***

```
mese = input('Inserisci il tuo mese di nascita (valore numerico): ');

switch mese
    case 1
        meseStr = 'Gennaio';
    case 2
        meseStr = 'Febbraio';
     case 3
        meseStr = 'Marzo';
    ...
    ...
    ...
    case 12
        meseStr = 'Dicembre';
    otherwise
        disp('Errore: Il valore deve essere compreso tra 1 e 12')
end

disp(['Ciao, sei nato il mese di ', meseStr])
```

Costrutto SWITCH-CASE (3)

- ***Esempio 2 – Più espressioni per blocco case***

```
cifra = input('Inserisci cifra > 0 : ');

switch cifra
    case {2,4,6,8}
        disp('Cifra PARI');
    case {1,3,5,7,9}
        disp('Cifra DISPARI');
    otherwise
        disp('Errore: La cifra deve essere compresa tra 1 e 9')
end
```

Strutture iterative in MATLAB (1)

- Se una sezione di codice può essere potenzialmente ***ripetuta***, essa potrebbe essere definita come ***ciclo*** **(*loop*)**

- Due diverse tipologie di *ciclo* in MATLAB:
  - *For*
  - *While*

Strutture iterative in MATLAB (2)

For o While?

- Il ciclo **for** viene usato quando il **numero di ripetizioni (iterazioni) è noto a priori**
- Qualora invece **non fosse noto il numero di iterazioni**, è possibile utilizzare il ciclo **while**

Ciclo FOR (1)

- ***Sintassi***

```
for variabile = valori_array
   blocco_istruzioni
end
```

Ciclo FOR (1)

- ***Sintassi***

```
for variabile = valori_array
   blocco_istruzioni
end
```

- Il ciclo `for` risulta essere molto utile anche per la manipolazione di array e di matrici

Ciclo FOR (2)

- ***Esempio 1***

```
for i = [1:1:10]
    disp(i)
end
```

Ciclo FOR (2)

- ***Esempio 1***

```
for i = [1:1:10]
    disp(i)
end
```

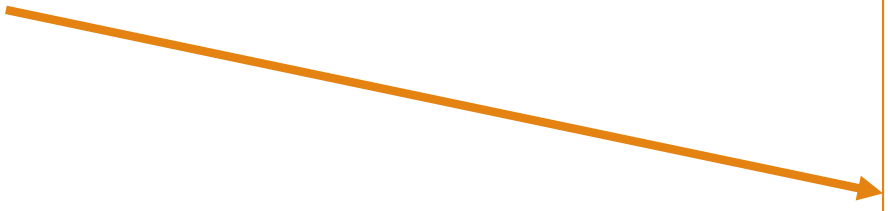
- **[1:1:10]** equivale al vettore [1 2 3 4 5 6 7 8 9 10]

Ciclo FOR (2)

- ***Esempio 1***

```
for i = [1:1:10]
    disp(i)
end
```

- ***Output***



```
     1
     2
     3
     4
     5
     6
     7
     8
     9
    10
```

Ciclo FOR (3)

- ***Esempio 2***

```
for i = 2:2:10
    disp(i)
end
```

Ciclo FOR (3)

- ***Esempio 2***

```
for i = 2:2:10
    disp(i)
end
```

- ***Output***
  - Numeri pari da 2 a 10

```
     2

     4

     6

     8

    10
```

Ciclo FOR (4)

- ***Esempio 3***
  - Funzione che prende in input un array `x` e restituisce la somma

```
function [somma] = somma_array(x)
    s = 0
    for i = 1:length(x)
        s = s + x(i);
    end
    somma = s;
end
```

Ciclo FOR (4)

- ***Esempio 3***
  - Funzione che prende in input un array `x` e restituisce la somma

```
function [somma] = somma_array(x)
    s = 0
    for i = 1:length(x)
        s = s + x(i);
    end
    somma = s;
end
```

→ restituisce la dimensione di `x`

Ciclo FOR (4)

- ***Esempio 3***
  - Funzione che prende in input un array `x` e restituisce la somma

```
function [somma] = somma_array(x)
    s = 0
    for i = 1:length(x)
        s = s + x(i);
    end
    somma = s;
end
```

  - **Esempi d'uso**

```
>> mio_array = [200 -1 5 24 8];
>> somma_mio_array = somma_array(mio_array)
somma_mio_array =
   236

>> somma = somma_array(1:10)
somma =
    55
```

Ciclo FOR (5)

- ***Esempio 4***
  - Funzione che prende in input un array `x` e restituisce la somma degli elementi con indice pari e la somma degli elementi con indice dispari

```
function [somma_pari, somma_disp] = somma_pari_dispari(x)
    somma_pari = 0;
    somma_disp = 0;

    for i = 1:1:length(x)
        if (mod(i, 2) == 0) % indice pari
            somma_pari = somma_pari + x(i);
        else % indice dispari
            somma_disp = somma_disp + x(i);
        end
    end
end
```

NOTA: I commenti riportati in `verde` saranno **ignorati da MATLAB**. Essi sono di aiuto solo a chi scrive/vede il codice sorgente della function

Ciclo FOR (5)

- ***Esempio 4***
  - Funzione che prende in input un array `x` e restituisce la somma degli elementi con indice pari e la somma degli elementi con indice dispari

```
function [somma_pari, somma_disp] =
somma_pari_dispari(x)
    somma_pari = 0;
    somma_disp = 0;

    for i = 1:1:length(x)
        if (mod(i, 2) == 0) % indice
            somma_pari = somma_pari
        else % indice dispari
            somma_disp = somma_disp
        end
    end
end
```

NOTA: I commenti riportati in verde s
da MATLAB. Essi sono di aiuto solo a ch
codice sorgente della function

Esempi d'uso

```
>> [sommap, sommad] = somma_pari_dispari([1 2 3 4 5])
sommap =
     6
sommad =
     9

>> [sommap, sommad] = somma_pari_dispari([1 3 5 7])
sommap =
    10
sommad =
     6
```

Ciclo FOR (6)

- ***Esempio 5***
  - Funzione che prende in input un array `x` e restituisce l'elemento con valore massimo

```
function [massimo] = massimo(x)
    massimo = x(1);
    for i = 2:length(x)
        if x(i) > massimo
            massimo = x(i);
        end
    end
end
```

Ciclo FOR (6)

- ***Esempio 5***
  - Funzione che prende in input un array `x` e restituisce l'elemento con valore massimo

```
function [massimo] = massimo(x)
    massimo = x(1);
    for i = 2:length(x)
        if x(i) > massimo
            massimo = x(i);
        end
    end
end
```

Esempi d'uso

```
>> max = massimo([10 20 40 500 35])
max =
   500
>> max = massimo([-8 -1 -25 -10 -45])
max =
    -1
>> max = massimo([1.5 2.25 1/1000 2/3 8/29])
max =
    2.2500
```

Ciclo FOR (7)

- ***Esempio 6***
  - Funzione che prende in input due array `a` e `b` (entrambi della stessa lunghezza) e restituisce un array `c`, in cui ogni elemento `c(i)` contiene il valore dell'elemento massimo tra `a(i)` e `b(i)`

  - ***Esempio Input/output funzione:***
    - **INPUT**
      - `a = [ 29 14 44 58 ]`
      - `b = [ 11 49  2 54 ]`
    - **OUTPUT**
      - `c = [ 29 49 44 58 ]`

  - ***Suggerimento***: possiamo utilizzare la funzione `massimo`, che abbiamo visto in precedenza

Ciclo FOR (7)

- ***Esempio 6***
  - Funzione che prende in input due array `a` e `b` (entrambi della stessa lunghezza) e restituisca un array `c`, in cui ogni elemento `c(i)` contenga il valore dell'elemento massimo tra `a(i)` e `b(i)`

```
function [c] = massimi_valori(a, b)
    for i = 1:length(a)
        c(i) = massimo([a(i) b(i)]);
    end
end
```

Ciclo FOR (7)

- ***Esempio 6***
  - Funzione che prende in input due array `a` e `b` (entrambi della stessa lunghezza) e restituisce un array `c`, in cui ogni elemento `c(i)` contenga il valore dell'elemento massimo tra `a(i)` e `b(i)`

```
function [c] = massimi_valori(a, b)
    for i = 1:length(a)
        c(i) = massimo([a(i) b(i)]);
    end
end
```

Esempi d'uso

```
>> c = massimi_valori([2 3], [1, 8])
c =
     2     8

>> c = massimi_valori([ 29 14 44 58 ], [ 11 49  2  54])
c =
    29    49    44    58
```

Ciclo FOR (8)

- ***Esempio 7***
  - Funzione che prende in input una matrice `A` e restituisce la somma degli elementi di `A`

  - ***Soluzione 1***

```
function [somma] = somma_matrice(A)
    [m n] = size(A);
    somma = 0;

    for r = 1:m
        for c = 1:n
            somma = somma + A(r, c);
        end
    end
end
```

Ciclo FOR (8)

- ***Esempio 7***
  - Funzione che prende in input una matrice A e restituisce la somma degli elementi di A

  - ***Soluzione 2 (utilizzo di una funzione vista precedentemente)***

```
function [somma] = somma_matrice_soluzione2(A)
    [m n] = size(A);
    somma = 0;

    for r = 1:m
        riga_A = A(r, :); % estraggo r-esima riga di A
        somma = somma + somma_array(riga_A);
    end
end
```

Ciclo FOR (9)

- ***Esempio 8***
  - Funzione che prende in input una matrice `vs` (***v**oti **s**tudenti*) e un array `cfu` (*CFU esami*), rappresentati come segue:

`vs`

Studenti/Voti esame	Esame 1	Esame 2	Esame 3	Esame 4	Esame 5
Studente Matricola 1	28	25	30	23	19
Studente Matricola 2	24	27	28	21	24
Studente Matricola 3	25	25	19	18	22
Studente Matricola 4	21	30	30	22	30

`cfu`

	Esame 1	Esame 2	Esame 3	Esame 4	Esame 5
CFU	6	3	2	9	4

  - La funzione restituisce, in output, un array contenente la **media ponderata** di ogni studente

Ciclo FOR (9)

- ***Esempio 8***
  - Funzione che prende in input una matrice `vs` (*voti studenti*) e un array `ce` (*CFU esami*), rappresentati come segue:

`vs`

Studenti/Voti esame	Esame 1	Esame 2	Esame 3	Esame 4	Esame 5
Studente Matricola 1	28	25	30	23	19
Studente Matricola 2	24	27	28	21	24
Studente Matricola 3	25	25	19	18	22

Media Ponderata:

$$\frac{(voto\_esame\_1*cfu1) + (voto\_esame\_2*cfu2) + \ldots + (voto\_esame\_N*cfuN)}{(cfu1+cfu2+\ldots+cfuN)}$$

  - La funzione restituisce, in output, un array contenente la **media ponderata** di ogni studente

vs

Studenti/Voti esame	Esame 1	Esame 2	Esame 3	Esame 4	Esame 5
Studente Matricola 1	28	25	30	23	19
Studente Matricola 2	24	27	28	21	24
Studente Matricola 3	25	25	19	18	22
Studente Matricola 4	21	30	30	22	30

cfu

	Esame 1	Esame 2	Esame 3	Esame 4	Esame 5
CFU	6	3	2	9	4

- ***Esempio 8 – Soluzione step-by-step (1)***
  - ***Riduzione del problema:*** Riduciamo il problema principale ad uno più semplice:
    - Consideriamo solo una riga della matrice `vs`

vs

Studenti/Voti esame	Esame 1	Esame 2	Esame 3	Esame 4	Esame 5
Studente Matricola 1	28	25	30	23	19
Studente Matricola 2	24	27	28	21	24
Studente Matricola 3	25	25	19	18	22
Studente Matricola 4	21	30	30	22	30

cfu

	Esame 1	Esame 2	Esame 3	Esame 4	Esame 5
CFU	6	3	2	9	4

- ***Esempio 8 – Soluzione step-by-step (1)***
  - ***Riduzione del problema:*** Riduciamo il problema principale ad uno più semplice:
    - Consideriamo solo una riga della matrice `vs`
  - La semplificazione consiste nel fatto che ora dobbiamo **calcolare la media ponderata** di **un solo studente**
    - Scriviamo quindi una funzione che esegua ciò, che prenderà in input due array: `riga_vs` (una riga di `vs`) e `cfu`

vs

Studenti/Voti esame	Esame 1	Esame 2	Esame 3	Esame 4	Esame 5
Studente Matricola 1	28	25	30	23	19
Studente Matricola 2	24	27	28	21	24
Studente Matricola 3	25	25	19	18	22
Studente Matricola 4	21	30	30	22	30

cfu

	Esame 1	Esame 2	Esame 3	Esame 4	Esame 5
CFU	6	3	2	9	4

- ***Esempio 8 – Soluzione step-by-step (1)***
  - ***Riduzione del problema:*** Riduciamo il problema principale ad uno più semplice:
    - Consideriamo solo una riga della matrice `vs`
  - La semplificazione consiste nel fatto che ora dobbiamo calcolare la media ponderata di un solo studente
    - Scriviamo quindi una funzione che esegua ciò, che prenderà in input due array: `riga_vs` (una riga di `vs`) e `cfu`
    - Affrontiamo quindi il sotto-problema del **calcolo della media Ponderata di un singolo studente**

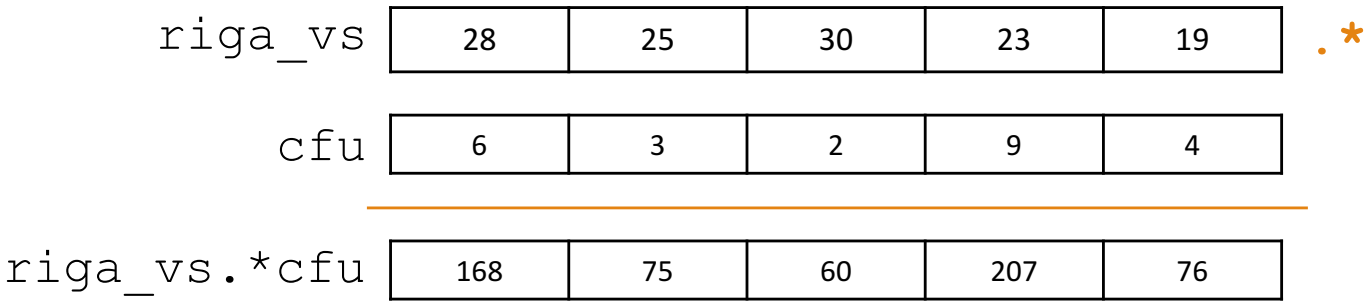
Ciclo FOR (9)

- ***Esempio 8 – Soluzione step-by-step (2)***
  - ***<u>Sotto-problema:</u> Calcolo Media Ponderata (singolo studente)***

`riga_vs`	28	25	30	23	19
`cfu`	6	3	2	9	4

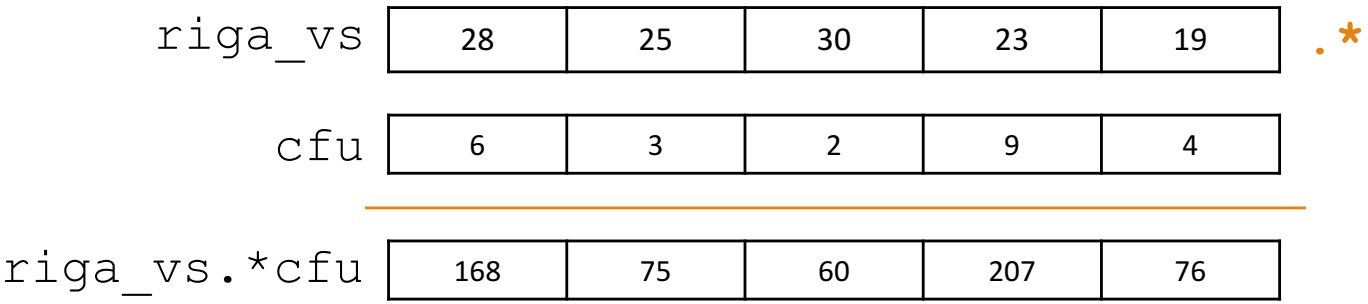
Ciclo FOR (9)

- ***Esempio 8 – Soluzione step-by-step (2)***
  - ***Sotto-problema:*** ***Calcolo Media Ponderata (singolo studente)***
    - ***Idea:*** Sfruttiamo la moltiplicazione elemento per elemento (.*)



Ciclo FOR (9)

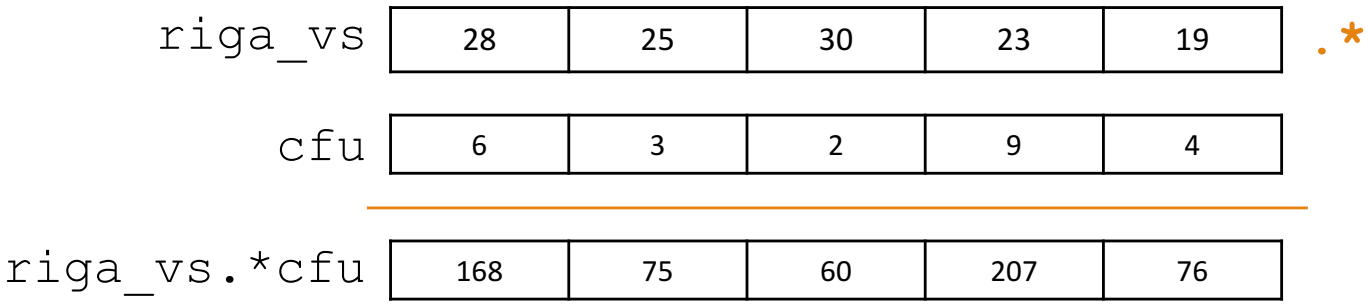
- ***Esempio 8 – Soluzione step-by-step (2)***
  - ***Sotto-problema:*** ***Calcolo Media Ponderata (singolo studente)***
    - ***Idea:*** Sfruttiamo la moltiplicazione elemento per elemento (.*)



- Per calcolare la media ponderata, ora necessitiamo solo di **due somme**:
  - La **somma** degli elementi dell'**array `riga_vs.*cfu`**
    - Lo otteniamo con → `sum(riga_vs.*cfu)`
  - La **somma dei CFU** e possiamo ottenerla tramite → `sum(cfu)`

Ciclo FOR (9)

- ***Esempio 8 – Soluzione step-by-step (2)***
  - ***Sotto-problema:*** *Calcolo Media Ponderata (singolo studente)*
    - ***Idea:*** Sfruttiamo la moltiplicazione elemento per elemento (.*)



- Otteniamo la **media ponderata** calcolando:
  - `sum(riga_vs.*cfu) / sum(cfu)`

Ciclo FOR (9)

- ***Esempio 8 – Soluzione step-by-step (3)***
  - *Funzione MATLAB media ponderata singolo studente*

```
function [media_pond_stud] = media_ponderata_studente(riga_vs, cfu)
    media_pond_stud = sum(riga_vs.*cfu) / sum(cfu);
end
```

vs

Studenti/Voti esame	Esame 1	Esame 2	Esame 3	Esame 4	Esame 5
Studente Matricola 1	28	25	30	23	19
Studente Matricola 2	24	27	28	21	24
Studente Matricola 3	25	25	19	18	22
Studente Matricola 4	21	30	30	22	30

cfu

	Esame 1	Esame 2	Esame 3	Esame 4	Esame 5
CFU	6	3	2	9	4

- ***Esempio 8 – Soluzione step-by-step (4)***
  - *Funzione media ponderata di tutti gli studente*
    - Sfrutta la funzione `media_ponderata_studente` (slide precedente)

```
function [mp] = media_ponderata(vs, cfu)
    [nrighe_vs, ncolonne_vs] = size(vs);

    for i = 1:nrighe_vs
        riga_vs = vs(i, :); % estraggo l'i-esima riga di vs

        mp(i) = media_ponderata_studente(riga_vs, cfu);
    end
    mp = mp'; % viene fatta la trasposta per ottenere un vettore colonna
end
```

vs

Studenti/Voti esame	Esame 1	Esame 2	Esame 3	Esame 4	Esame 5
Studente Matricola 1	28	25	30	23	19
Studente Matricola 2	24	27	28	21	24
Studente Matricola 3	25	25	19	18	22
Studente Matricola 4	21	30	30	22	30

```
24.4167
23.5833
21.3750
24.7500
```

cfu

	Esame 1	Esame 2	Esame 3	Esame 4	Esame 5
CFU	6	3	2	9	4

- ***Esempio 8 – Soluzione step-by-step (4)***
  - *Funzione media ponderata di tutti gli studente* (***Esempio d'uso***)

```
>> vs = [28 25 30 23 19; 24 27 28 21 24; 25 25 19 18 22; 21 30 30 22 30];
>> cfu = [6 3 2 9 4];
>> mp = media_ponderata(vs, cfu)

mp =

   24.4167
   23.5833
   21.3750
   24.7500
```

Ciclo WHILE (1)

- ***Sintassi***

```
while condizione
    blocco_istruzioni
end
```

- Finché la `condizione` del `while` è verificata (quindi risulta essere vera), le istruzioni contenute nel corpo del ciclo `blocco_istruzioni` verranno eseguite
  - Al contrario appena la condizione non sarà più verificata (risulterà falsa), il ciclo terminerà

- Come detto precedentemente, Il ciclo `while` è estremamente utile quando non sono note a priori il numero di iterazioni che dovranno essere eseguite

Ciclo WHILE (2)

- ***Esercizio 1***

```
x = 20;
while x < 50
    disp(x);
    x = x + 3;
end
```

Output

```
    20
    23
    26
    29
    32
    35
    38
    41
    44
    47
```

Ciclo WHILE (3)

- ***Esercizio 2***
  - Esegue il ciclo finché non viene immesso un valore negativo

```
somma = 0;
x = input('Inserisci x (> 0): ');
while x > 0
    somma = somma + x;
    x = input('Inserisci x (> 0): ');
end
disp(['somma: ', num2str(somma)]);
```

- ***Esempio d'uso***

```
Inserisci x (> 0): 10
Inserisci x (> 0): 9
Inserisci x (> 0): 8
Inserisci x (> 0): -1
somma: 27
```

Ciclo DO_WHILE

- *MATLAB non offre una sintassi per il ciclo do_while*
  - è comunque possibile ottenere un comportamento simile a quello del do_while impiegando un while come segue:

```
flag = true;
while flag
    % do stuff
    if ~condition
      flag = false;
    end
end
```

Istruzioni break e continue

- Le istruzioni **break** e **continue** possono essere usate, all'interno di un ciclo, per terminare o modificare l'esecuzione del ciclo stesso
  - L'istruzione **break** (*to break* → *rompere)* termina il ciclo anche se non si è raggiunta la fine dello stesso
  - L'istruzione **continue**, invece, ***salta* una iterazione** del ciclo ma fa si che il flusso rimanga comunque all'interno del ciclo

Riferimenti

- Capitolo 4
  - Paragrafi 2 [**Operatori relazionali**], 3 [**Operatori logici e funzioni**], 3 [**Istruzioni condizionali**], 4 [**Cicli for**], 5 [**Il ciclo while**], e 7 [**La struttura switch**].