Università di Salerno
Dipartimento di
Ingegneria Industriale

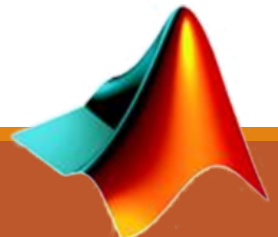
UNIVERSITÀ DEGLI STUDI DI SALERNO

Fondamenti di Informatica

Introduzione alla Programmazione in MATLAB: Parte 3 (Strutture Iterative)

Prof. Arcangelo Castiglione

A.A. 2016/17



MATLAB

Introduzione alla programmazione in MATLAB: Parte 3

OUTLINE

- Ricapitolazione
- Strutture Iterative
  - Costrutto FOR
  - Costrutto WHILE
  - Istruzioni break e continue

Breve Ricapitolazione – 1/5

- Nella scorsa lezione abbiamo visto che MATLAB fornisce tre strutture di controllo fondamentali

Breve Ricapitolazione – 2/5

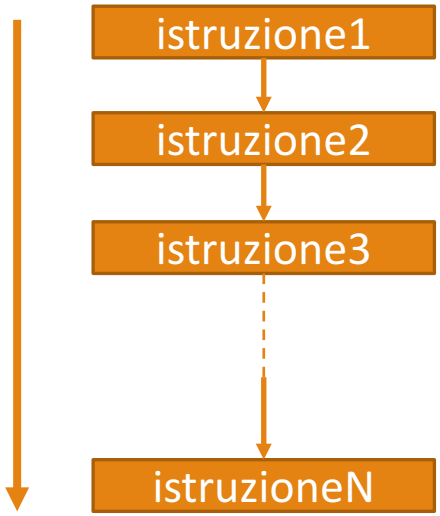
- Nella scorsa lezione abbiamo visto che MATLAB fornisce tre strutture di controllo fondamentali

- Sequenza

Breve Ricapitolazione – 3/5

- Nella scorsa lezione abbiamo visto che MATLAB fornisce tre strutture di controllo fondamentali

- **Sequenza**



Breve Ricapitolazione – 4/5

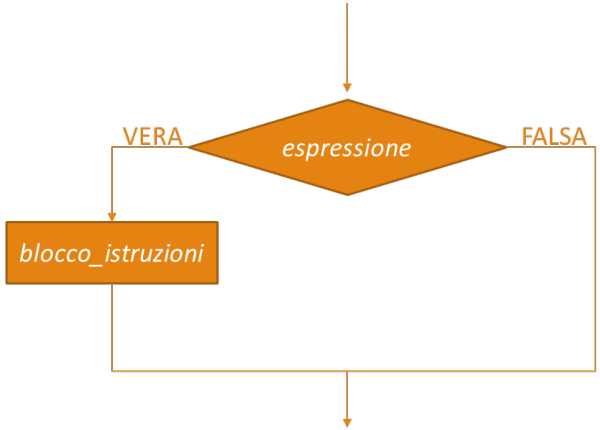
- Nella scorsa lezione abbiamo visto che MATLAB fornisce tre strutture di controllo fondamentali

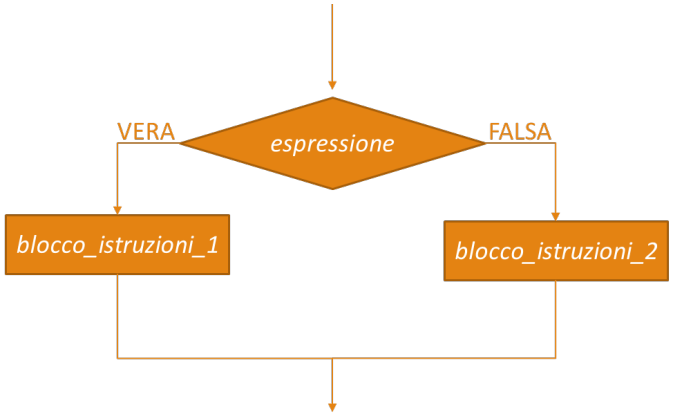
- Sequenza
- Selezione

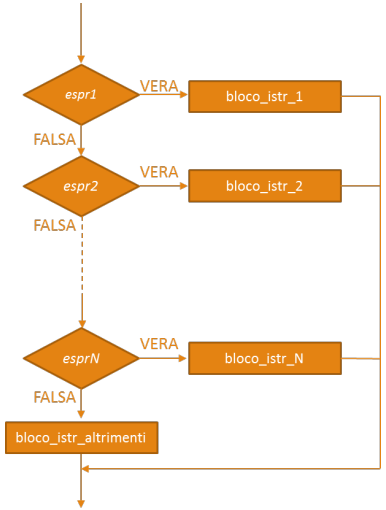
Breve Ricapitolazione – 5/5

- Nella scorsa lezione abbiamo visto che MATLAB fornisce tre strutture di controllo fondamentali

- Sequenza
- **Selezione**







Strutture Iterative – 1/6

- Nella scorsa lezione abbiamo visto che MATLAB fornisce tre strutture di controllo fondamentali

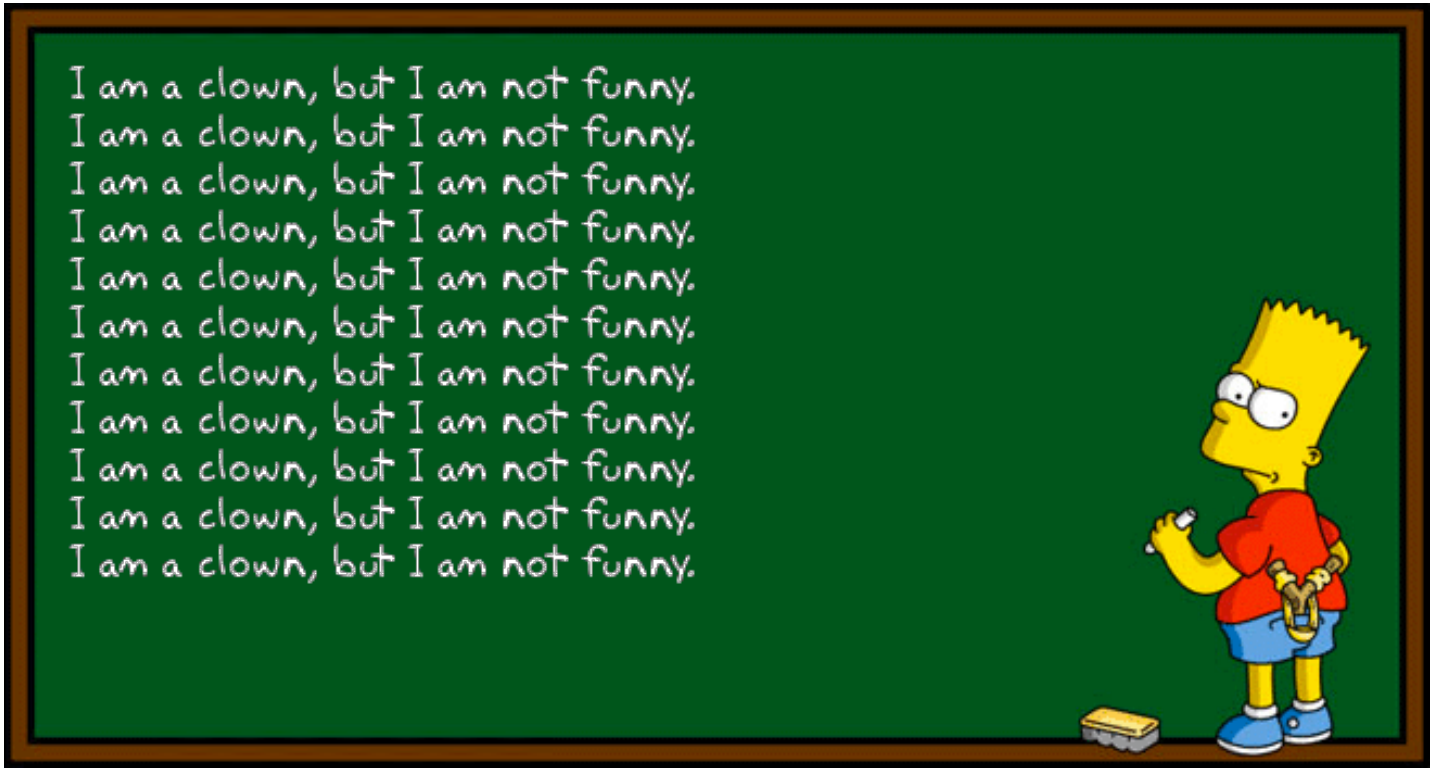
- Sequenza
- Selezione
- **Iterazione**

Strutture Iterative – 2/6

- Perché abbiamo bisogno dell'**Iterazione**?

Strutture Iterative – 3/6

- Per ripetere azioni! Ad es., per stampare a video qualcosa un certo numero di volte, cercare qualcosa all'interno di un testo, etc



Strutture Iterative – 4/6

- Possibile modo per farlo…

```
>> disp('I am a clown, but I am not funny.');
>> disp('I am a clown, but I am not funny.');
>> disp('I am a clown, but I am not funny.');
>> disp('I am a clown, but I am not funny.');
>> disp('I am a clown, but I am not funny.');
>> disp('I am a clown, but I am not funny.');
>> disp('I am a clown, but I am not funny.');
>> disp('I am a clown, but I am not funny.');
>> disp('I am a clown, but I am not funny.');
…
>> disp('I am a clown, but I am not funny.');
>> disp('I am a clown, but I am not funny.');
>> disp('I am a clown, but I am not funny.');
>> disp('I am a clown, but I am not funny.');
>> disp('I am a clown, but I am not funny.');
>> disp('I am a clown, but I am not funny.');
>> disp('I am a clown, but I am not funny.');
…
```

Per un certo numero di volte

Strutture Iterative – 5/6

Strutture Iterative – 6/6

- Se una **sezione di codice** può essere **potenzialmente ripetuta**, essa deve essere inserita in una **struttura iterativa (o ciclo)**
- Un ciclo (o loop) è una struttura di controllo che permette di ripetere l'esecuzione di istruzioni per un certo numero di volte
  - Ogni ripetizione del ciclo è detta **iterazione o (passo)**
- MATLAB fornisce di due tipologie di struttura iterativa
  - Ciclo `for`
    - Utilizzato quando il **numero di iterazioni** è **noto a priori**
  - Ciclo `while`
    - Utilizzato quando il **numero di iterazioni non** è **noto a priori**
      - Il ciclo termina quando è soddisfatta una particolare condizione

Ciclo FOR – 1/10

- **Ciclo `for` (Sintassi MATLAB)**

```
for variabile = m:s:n
   blocco_istruzioni
end
```

- L'espressione `m:s:n` assegna il valore iniziale `m` alla `variabile` di ciclo, che viene incrementata del valore `s` (detto step o incremento)
- Le istruzioni (`blocco_istruzioni`) vengono eseguite una sola volta per ogni iterazione, utilizzando il valore corrente della `variabile` di ciclo
- L'elaborazione continua finché la `variabile` di ciclo supera il valore finale per `n`
  - Di solito, la `variabile` di ciclo è uno scalare, ma può essere anche un vettore o una matrice

Ciclo FOR – 1/10

```
for k = 1 : 1 : 10
    x = k^2
end
```

M-File Script per calcolare l'elevazione al quadrato degli interi da 1 a 10 (potenza.m)

```
>> potenza
x =
    1
x =
    4
x =
    9
x =
    16
x =
    25
x =
    36
x =
    49
x =
    64
x =
    81
x =
    100
```

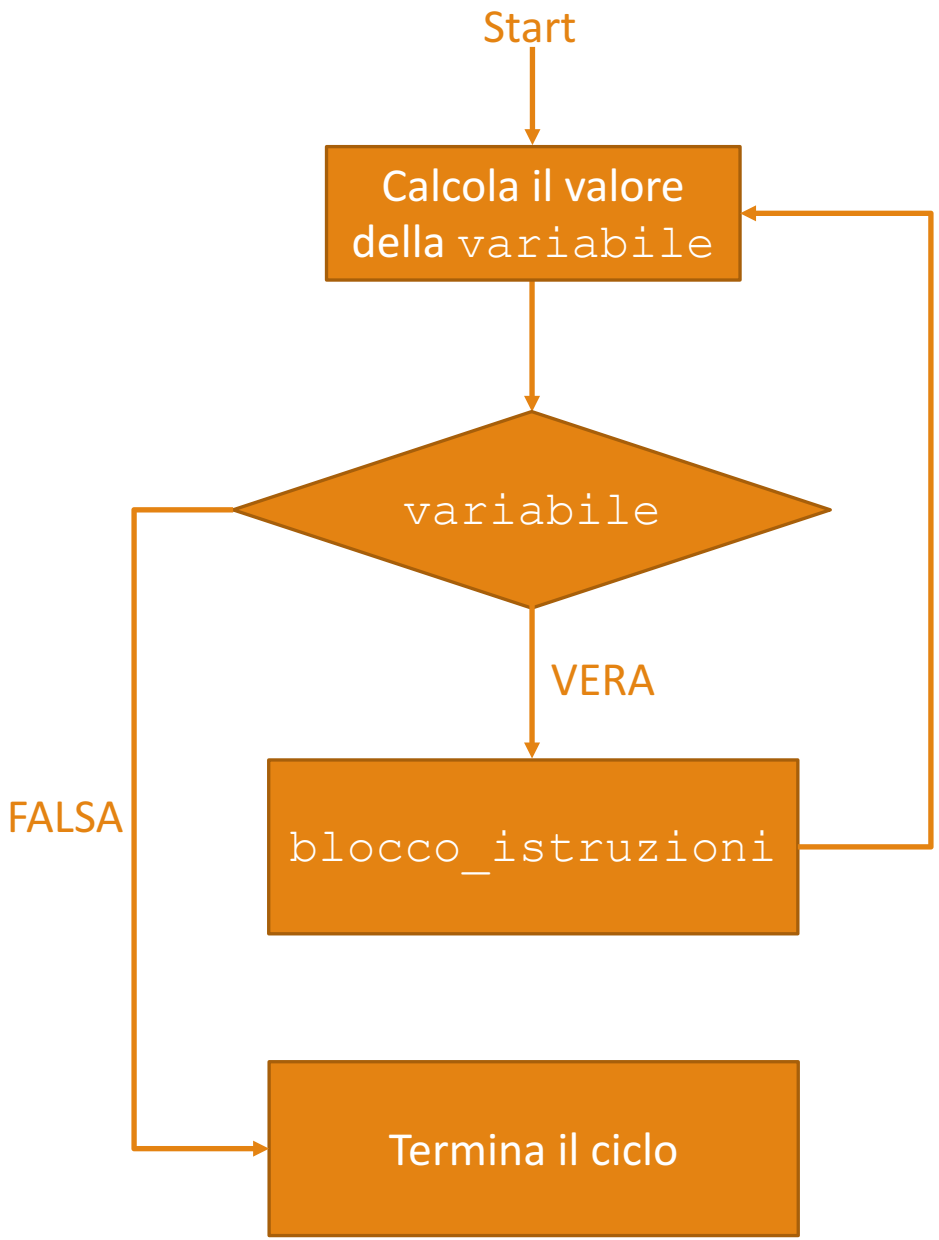
Esecuzione

Ciclo FOR – 1/10

L'istruzione `x = k^2` viene eseguita una sola volta per ogni iterazione, utilizzando il valore corrente della **variabile di ciclo** `k`

```
for k = 1 : 1 : 10
    x = k^2
end
```

M-File Script per calcolare l'elevazione al quadrato degli interi da 1 a 10 (potenza.m)


Start
Calcola il valore della variabile
variabile
VERA
FALSA
blocco_istruzioni
Termina il ciclo

Ciclo FOR – 1/10

- **Ciclo `for` (Sintassi MATLAB)**

```
for variabile = m:s:n
   blocco_istruzioni
end
```

- Si noti che ogni `for` deve essere associato ad una clausola `end`
  - `end` indica la fine delle istruzioni da eseguire all'interno del ciclo

Ciclo FOR – 1/10

- **Ciclo `for` (Sintassi MATLAB)**

```
for variabile = m:s:n
   blocco_istruzioni
end
```

- Fra la clausola `for` ed il nome della `variabile` di ciclo deve essere inserito uno spazio

Ciclo FOR – 1/10

- **Ciclo `for` (Sintassi MATLAB)**

```
for variabile = m:s:n
    blocco_istruzioni
end
```

- È buona norma indentare (rientrare) verso destra le istruzioni del ciclo `for` e la corrispondente clausola `end`, per mettere in evidenza quali istruzioni appartengono al ciclo

Ciclo FOR – 2/10

- **Ciclo `for` (Sintassi alternativa MATLAB)**

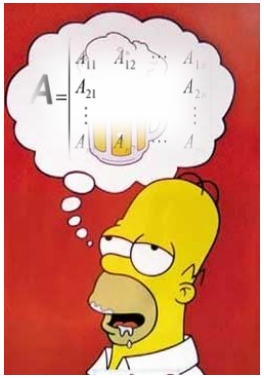
```
for variabile = valori_array
   blocco_istruzioni
end
```

Ciclo FOR – 2/10

- **Ciclo `for` (Usare array e matrici come indice di ciclo)**

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

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



Ciclo FOR – 3/10

- ***Esempio 1***

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

Ciclo FOR – 3/10

- ***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 – 3/10

- ***Esempio 1***

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

- ***Output***

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

Ciclo FOR – 4/10

- ***Esempio 2***

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

Ciclo FOR – 4/10

- ***Esempio 2***

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

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

```
     2

     4

     6

     8

    10
```

Ciclo FOR – 5/10

- ***Esempio 3***
  - Funzione che prende in input un array `x` e restituisce in output la somma degli elementi che esso contiene

Ciclo FOR – 5/10

- ***Esempio 3***
  - Funzione che prende in input un array `x` e restituisce in output la somma degli elementi che esso contiene

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

Ciclo FOR – 5/10

- ***Esempio 3***
  - Funzione che prende in input un array `x` e restituisce in output la somma degli elementi che esso contiene

```
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 – 5/10

- ***Esempio 3***

```
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 – 6/10

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

Ciclo FOR – 6/10

- ***Esempio 4***
  - Funzione che prende in input un array `x` e restituisce in output 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
```

help **mod**

N.B.: I commenti (riportati in `verde`) saranno **ignorati da MATLAB**. Essi sono di aiuto solo a chi scrive/legge il codice sorgente della funzione

Ciclo FOR – 6/10

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

N.B. i vari livelli di indentazione

```
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
```

N.B.: I commenti (riportati in `verde`) saranno **ignorati da MATLAB**. Essi sono di aiuto solo a chi scrive/legge il codice sorgente della funzione

Ciclo FOR – 6/10

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

```
function [somma_pari, somma_disp] = somma_p
    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
```

N.B. i vari livelli di indentazione

- **N.B.**
  - È possibile annidare i cicli `for` e le istruzioni condizionali
  - Si noti che `for` ed `if` devono essere sempre accompagnati dalla relativa clausola `end`

N.B.: I commenti (riportati in `verde`) saranno **ignorati da MATLAB**. Essi sono di aiuto solo a chi scrive/legge il codice sorgente della funzione

Ciclo FOR – 6/10

- ***Esempio 4***
  - Funzione che prende in input un array `x` e restituisce in output 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) % indic
            somma_pari = somma_pari
        else % indice dispari
            somma_disp = somma_disp
        end
    end
end
```

N.B.: I commenti (riportati in ver
ignorati da MATLAB. Essi sono di aiu
scrive/legge il codice sorgente della funz

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 – 7/10

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

Ciclo FOR – 7/10

- ***Esempio 5***
  - Funzione che prende in input un array `x` e restituisce in output 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 – 7/10

- ***Esempio 5***
  - Funzione che prende in input un array `x` e restituisce in output 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 – 8/10

- ***Esempio 6***
  - Funzione che prende in input due array `a` e `b` (entrambi della stessa lunghezza) e restituisce in output 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 definito in precedenza

Ciclo FOR – 8/10

- ***Esempio 6***
  - Funzione che prende in input due array `a` e `b` (entrambi della stessa lunghezza) e restituisce in output un array `c`, in cui ogni elemento `c(i)` contiene 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 – 8/10

- ***Esempio 6***
  - Funzione che prende in input due array `a` e `b` (entrambi della stessa lunghezza) e restituisce in output un array `c`, in cui ogni elemento `c(i)` contiene 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 – 9/10

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

Ciclo FOR – 9/10

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

  - ***Soluzione 1***

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

    for r = 1:m %indicizza le righe della matrice
        for c = 1:n %indicizza le colonne della matrice
            somma = somma + A(r, c);
        end
    end
end
```

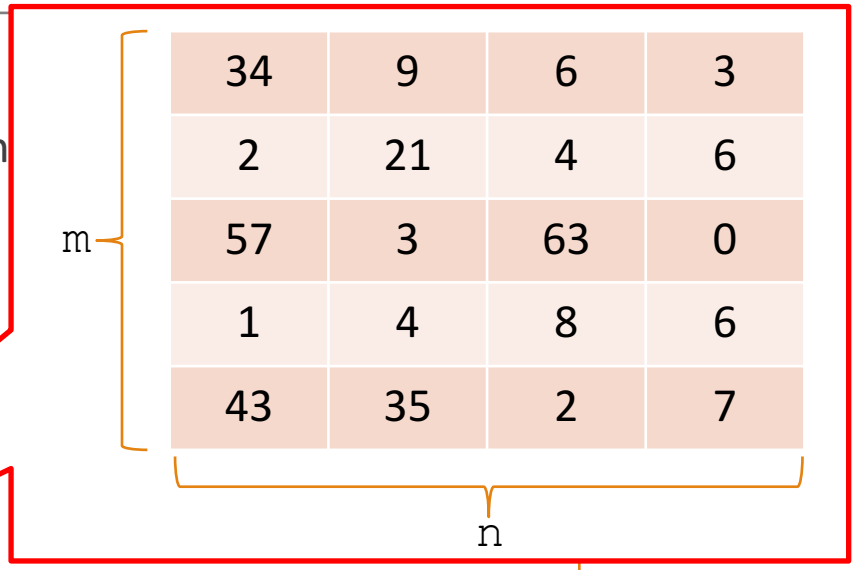
Ciclo FOR – 9/10

- ***Esempio 7***
  - Funzione che prende in input una m
    somma degli elementi di A

  - ***Soluzione 1***

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

    for r = 1:m %indicizza le righe della matrice
        for c = 1:n %indicizza le colonne della matrice
            somma = somma + A(r, c);
        end
    end
end
```



Ciclo FOR – 9/10

- ***Esempio 7***
  - Funzione che prende in input una matrice `A` e restituisce in output la somma degli elementi di `A`
  - ***Soluzione 2 (Mediante la funzione `somma_array` definita 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 – 10/10

- ***Esempio 8***
  - Funzione che prende in input una matrice `vs` (***voti studenti***) 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 – 10/10

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 (divide et impera)
    - 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 (divide et impera)
    - 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 effettui questa operazione. Tale funzione deve prendere 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 (divide et impera)
    - 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 effettui questa operazione. Tale funzione deve prendere 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 – 10/10

- ***Esempio 8 – Soluzione Step-By-Step (2)***
  - ***Sotto-problema:*** *Calcolo Media Ponderata (singolo studente)*

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

Ciclo FOR – 10/10

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

`riga_vs`	28	25	30	23	19	`.*`
`cfu`	6	3	2	9	4	
`riga_vs.*cfu`	168	75	60	207	76	

Ciclo FOR – 10/10

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

`riga_vs`	28	25	30	23	19	`.*`
`cfu`	6	3	2	9	4	
`riga_vs.*cfu`	168	75	60	207	76	

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

Ciclo FOR – 10/10

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

`riga_vs`	28	25	30	23	19	`.*`
`cfu`	6	3	2	9	4	
`riga_vs.*cfu`	168	75	60	207	76	

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

Ciclo FOR – 10/10

- ***Esempio 8 – Soluzione Step-By-Step (3)***
  - *Funzione MATLAB per calcolare la media ponderata di un 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 MATLAB per calcolare la media ponderata di tutti gli studenti*
    - 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	24.4167
Studente Matricola 2	24	27	28	21	24	23.5833
Studente Matricola 3	25	25	19	18	22	21.3750
Studente Matricola 4	21	30	30	22	30	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 MATLAB per calcolare la media ponderata di tutti gli studenti* (***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 – 1/5

- **Ciclo `while` (Sintassi MATLAB)**

```
while condizione
   blocco_istruzioni
end
```

- Finché la `condizione` del `while` è verificata (risulta essere vera), le istruzioni contenute nel corpo del ciclo (`blocco_istruzioni`) verranno eseguite
  - D'altro canto, non appena la `condizione` non sarà più verificata (risulterà falsa), il ciclo terminerà
- Il ciclo `while` è estremamente utile quando non è noto a priori il numero di iterazioni che dovranno essere eseguite
  - Ad es., quando si vuole continuare un processo finché non è soddisfatta una particolare condizione

Ciclo WHILE – 2/5

- ***Esempio 1***

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

M-File Script

Output

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

Ciclo WHILE – 2/5

- ***Esempio 1***

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

M-File Script

```
41
44
47
```

- Affinché il ciclo `while` possa operare correttamente, devono verificarsi due condizioni
  - La variabile di ciclo (in questo caso `x`) deve avere un valore prima che venga eseguito il `while`

Ciclo WHILE – 2/5

- ***Esempio 1***

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

M-File Script

- Affinché il ciclo `while` possa operare correttamente, devono verificarsi due condizioni
  - La variabile di ciclo (in questo caso `x`) deve avere un valore prima che venga eseguito il `while`
  - La variabile di ciclo deve essere modificata in qualche modo dalle istruzioni del ciclo stesso

Ciclo WHILE – 2/5

- ***Esempio 1***

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

M-File Script

- Le istruzioni sono eseguite una volta per ogni iterazione del ciclo, utilizzando il valore corrente della variabile di ciclo (in questo caso `x`)

```
32
35
38
41
44
47
```

Ciclo WHILE – 3/5

- **Ciclo `while` (Sintassi MATLAB)**

```
while condizione
    blocco_istruzioni
end
```

- Ogni clausola `while` deve essere associata alla relativa clausola `end`

Ciclo WHILE – 3/5

- **Ciclo `while` (Sintassi MATLAB)**

```
while condizione
    blocco_istruzioni
end
```

- Le istruzioni del ciclo (`blocco_istruzioni`) dovrebbero essere rientrate verso destra per agevolare la lettura dei programmi

Ciclo WHILE – 4/5

- ***Esempio 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)]);
```

M-File Script

- ***Esempio d'uso***

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

Ciclo WHILE – 5/5

- Prestare estrema attenzione alla `condizione` di terminazione del ciclo
  - Potrebbe accadere che il ciclo non abbia mai termine (e questa cosa potrebbe essere voluta…)

Ciclo DO/WHILE
(Ciclo a Condizione Finale - Come Simularlo)

- MATLAB **non fornisce** in maniera diretta il ciclo `do/while`
  - È comunque possibile simulare il comportamento del `do/while` utilizzando il ciclo `while` nel modo seguente

```
continua_ciclo = true;
while continua_ciclo
   blocco_istruzioni;
   if ~condizione
      continua_ciclo = false;
   end
end
```

Esempio

```
continua_ciclo = true;
i = 1;

while continua_ciclo
   disp(i);

   if ~(i < 10)
      continua_ciclo = false;
   end

   i = i + 1;
end
```

break e continue – 1/4

- Le clausole `break` e `continue` possono essere usate all'interno di un ciclo per terminare o modificare l'esecuzione del ciclo stesso
  - La clausola `break` (*to break* → *rompere)* termina il ciclo, anche se non si è raggiunta la fine dello stesso)
    - Il flusso di esecuzione del programma "salta" alla prossima istruzione fuori dal ciclo

```
for i = 1 : 6
    disp(i)
    if i > 3
        break;
    end
end
```

```
1
2
3
4
```

Output

Come mai il ciclo stampa 4 numeri invece di 3?

break e continue – 1/4

- Le clausole `break` e `continue` possono essere usate all'interno di un ciclo per terminare o modificare l'esecuzione del ciclo stesso
  - La clausola `break` (*to break* → *rompere)* termina il ciclo, anche se non si è raggiunta la fine dello stesso)
    - Il flusso di esecuzione del programma "salta" alla prossima istruzione fuori dal ciclo

```
for i = 1 : 6
    disp(i)
    if i > 3
        break;
    end
end
```

- È possibile utilizzare un'istruzione `if` per "saltare fuori" dal ciclo, prima che la variabile di ciclo abbia raggiunto il suo valore finale
  - Per fare questo è possibile usare l'istruzione `break`, che termina un ciclo senza concludere l'intero programma

break e continue – 2/4

- Le clausole `break` e `continue` possono essere usate all'interno di un ciclo per terminare o modificare l'esecuzione del ciclo stesso
  - La clausola `continue` salta un'iterazione del ciclo, ma fa si che il flusso di esecuzione rimanga comunque all'interno del ciclo stesso
    - Il flusso di esecuzione del programma "salta" alla fine del ciclo (bypassando tutte le istruzioni presenti tra la clausola `continue` e quella `end`) e comincia con la prossima iterazione del ciclo

```
for i = 1 : 6
    if mod(i, 2) == 0
        continue;
    end
    disp(i)
end
```

```
1
3
5
```

Output

break e continue – 2/4

- Le clausole `break` e `continue` possono essere usate all'interno di un ciclo per terminare o modificare l'esecuzione del ciclo stesso
  - La clausola `continue` salta un'iterazione del ciclo, ma fa si che il flusso di esecuzione rimanga comunque all'interno del ciclo stesso
    - Il flusso di esecuzione del programma "salta" alla fine del ciclo (bypassando tutte le istruzioni presenti tra la clausola `continue` e quella `end`) e comincia con la prossima iter

```
for i = 1 : 6
    if mod(i, 2) == 0
        continue;
    end
    disp(i)
end
```

- Questa istruzione passa il controllo del programma alla successiva iterazione del ciclo `for` o `while` in cui figura, ignorando tutte le istruzione restanti nel corpo del ciclo
- Nei cicli annidati, l'istruzione `continue` passa il controllo del programma alla successiva iterazione del ciclo `for` o `while` in cui è contenuta

Esercizio

- Simulare una sequenza (**infinita**) di lanci di un dado a 6 facce
  - Se viene visualizzato 2 volte consecutive il numero 6, la simulazione si arresta
  - **Esempio di esecuzione:** 3 4 2 5 4 2 2 5 1 1 6 6

Riferimenti

- Capitolo 4
  - Paragrafi 5, 6 e 7