Università di Salerno
Dipartimento di Ingegneria Industriale

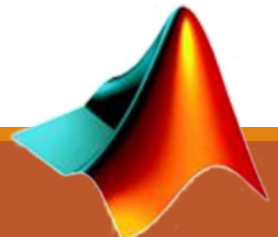
UNIVERSITÀ DEGLI STUDI DI SALERNO

Fondamenti di Informatica

Introduzione alla Programmazione in MATLAB: Parte 2 (Operatori Logico/Relazionali e Strutture Selettive)

Prof. Arcangelo Castiglione

A.A. 2016/17



MATLAB

Introduzione alla programmazione in MATLAB: Parte 2

OUTLINE

- Operatori Relazionali e Logici
- Strutture di Controllo in MATLAB
  - Strutture Selettive
    - Costrutto IF (e sue varianti)
    - Costrutto SWITCH-CASE

Operatori Relazionali – 1/6

- 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 – 2/6

- 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 – 2/6

- 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 è composto da **due simboli ==**, da non confondere con l'operatore di assegnazione, che è composto da un solo simbolo **=**

Operatori Relazionali – 2/6

- 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 – 3/6

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

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

Operatori Relazionali – 4/6

- ***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 – 5/6

- ***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 – 6/6

- ***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 – 1/4

- 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

Operatori Logici - Esempio 1

- 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 - Esercizio

- **Esercizio**

```
>> x = 1;
>> y = 2;
>> z = 3;
```

Espressione	Valore
4 && -5	
~0	
y > z && y > x	
~(y > z || y > x)	
(1 <= x) || ~((x * z / 3) < y) || (((y + z) – 4 * x) > 2)	
(y > 10) && ~(1 <= x + y) && (x^3 + y^3 > z ^2)	

Operatori Logici - Esempio 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 – 2/4
(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 – 3/4
(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];
>> mat_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

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

ans =
     10002   10005
```

Operatori Logici – 4/4
(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
```

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

Strutture Fondamentali di MATLAB – 1/3

- Analogamente ad altri linguaggi di programmazione, MATLAB fornisce tre **Strutture di Controllo** fondamentali
  - Sequenza
  - Selezione
  - Iterazione

Strutture Fondamentali di MATLAB – 2/3

- **Sequenza**
  - È definita dalla sequenza lessicografica delle istruzioni

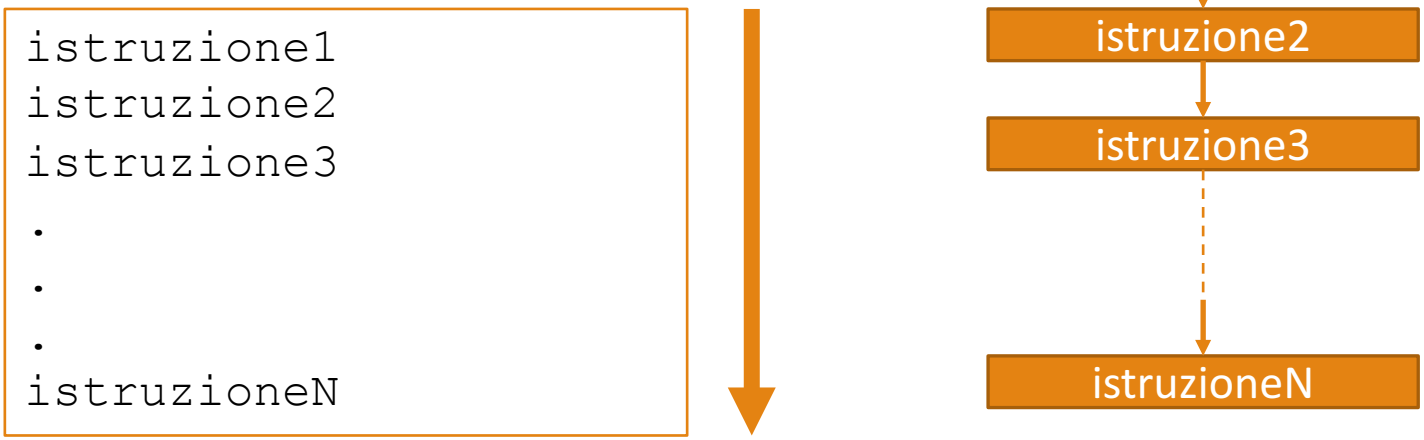
```
istruzione1
istruzione2
istruzione3
.
.
.
istruzioneN
```

Esecuzione TOP-DOWN
(alto verso il basso)

Strutture Fondamentali di MATLAB – 2/3

- **Sequenza**
  - È definita dalla sequenza lessicografica delle istruzioni

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



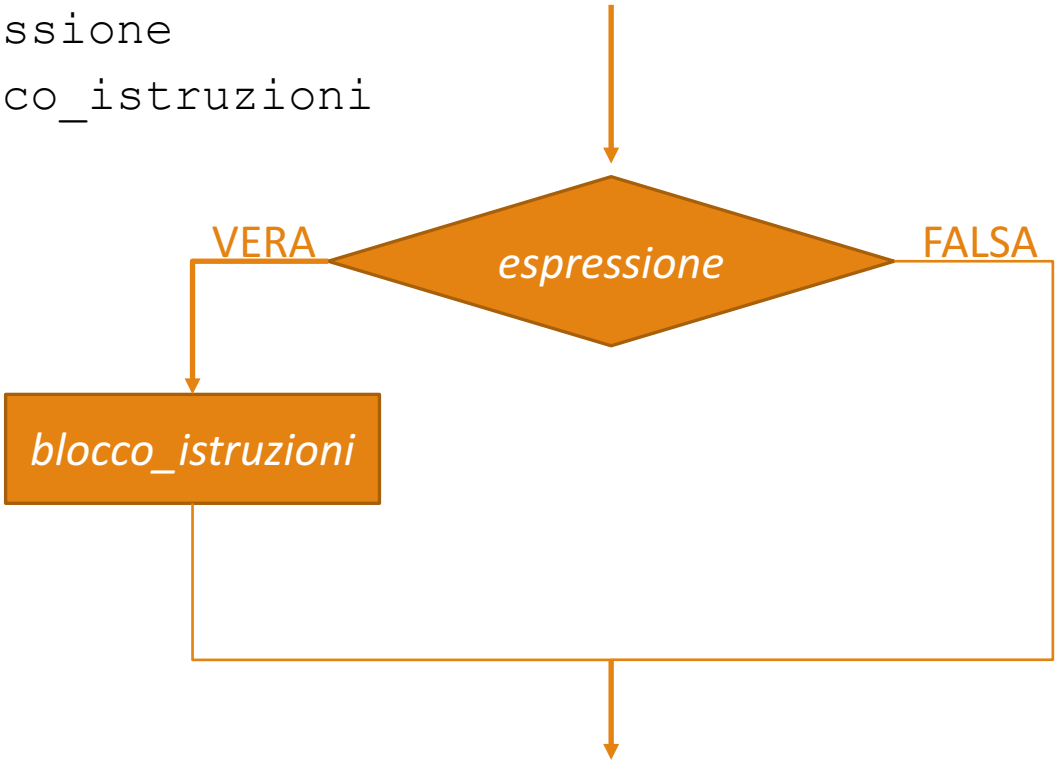
Strutture Fondamentali di MATLAB – 3/3

- **Selezione**
  - Le strutture selettive o decisionali
    - Ci consentono di scrivere programmi in grado di prendere delle decisioni
    - Sono implementate mediante *istruzioni condizionali*
    - Possono essere implementate in MATLAB mediante i costrutti *IF (e sue varianti)* e *SWITCH-CASE*
      - In generale, il costrutto *IF* contiene una o più clausole **`if`** (se), **`else`** (altrimenti) ed **`elseif`** (altrimenti se)

Costrutto IF – 1/6

- **Selezione Semplice (Sintassi MATLAB)**

```
if espressione
    blocco_istruzioni
end
```



Costrutto IF – 1/6

- **Selezione Semplice (Sintassi MATLAB)**

```
if espressione
    blocco_istruzioni
end
```

- **Se** l'`espressione` è vera (1) **allora** le istruzioni del blocco (`blocco_istruzioni`) sono eseguite
- **Se** l'`espressione` è falsa (0) **allora** le istruzioni del blocco (`blocco_istruzioni`) non sono eseguite

Costrutto IF – 1/6

- **Selezione Semplice (Sintassi MATLAB)**

```
if espressione
    blocco_istruzioni
end
```

- **N.B.** Ogni clausola `if` deve essere associata ad una clausola `end`
  - Quest'ultima clausola indica la fine delle istruzioni (`blocco_istruzioni`) che devono essere eseguite se l'`espressione` logica è vera
    - L'espressione logica può essere anche un'espressione composta

Costrutto IF – 1/6

- **Selezione Semplice (Sintassi MATLAB)**

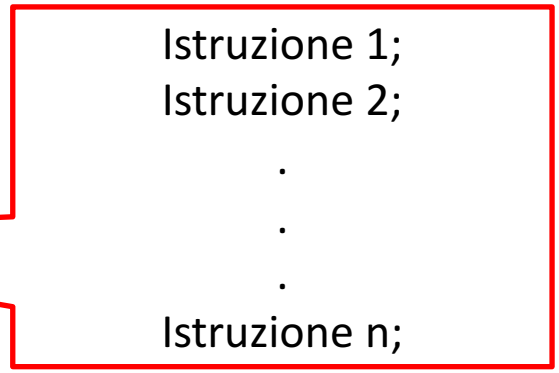
```
if espressione
    blocco_istruzioni
end
```

- Uno spazio deve separare la parola chiave `if` dall'`espressione` logica

Costrutto IF – 1/6

- **Selezione Semplice (Sintassi MATLAB)**

```
if espressione
    blocco_istruzioni
end
```



- Le istruzioni possono essere formate da una singola istruzione o da una serie di istruzioni
  - Di solito le istruzioni vengono rientrate verso destra (indentazione), per specificare che esse appartengono alla clausola `if` ed alla corrispondente clausola `end`

Costrutto IF – 1/6

- **Selezione Semplice (Sintassi MATLAB)**

```
if espressione
    blocco_istruzioni
end
```

- ***Esempio***

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

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

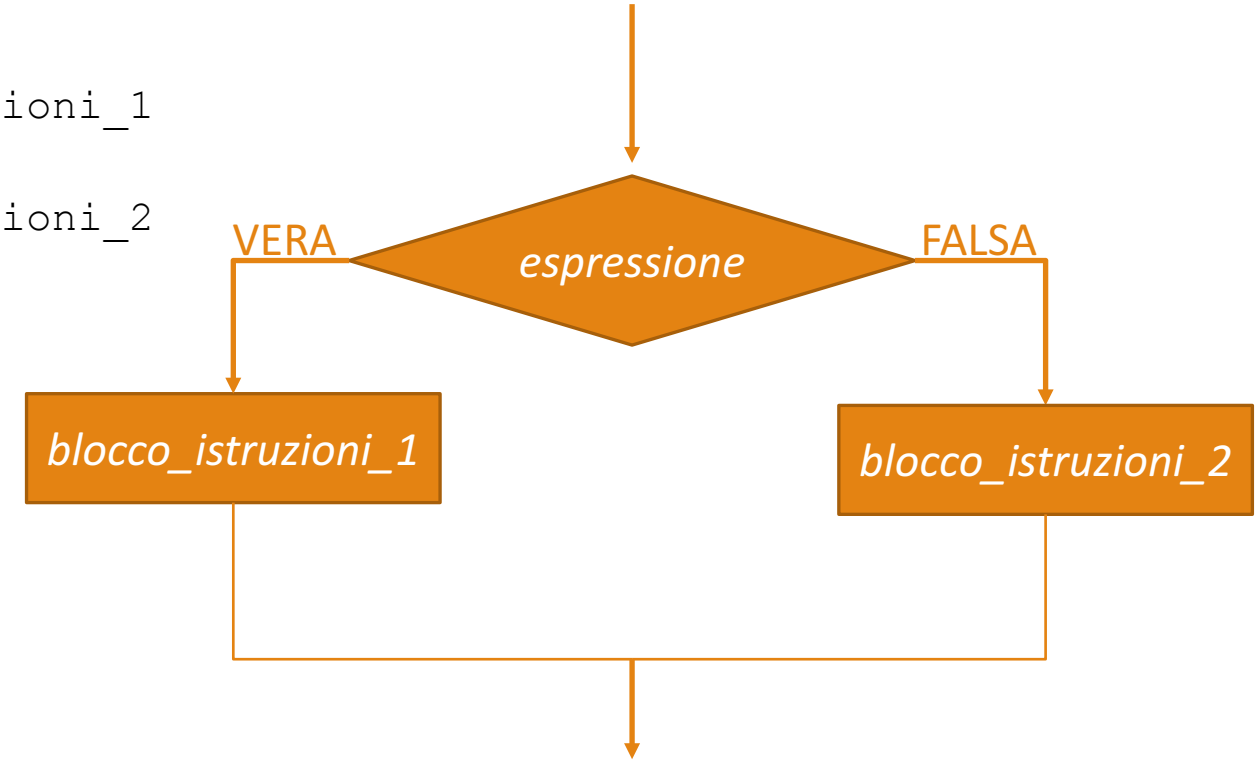
Costrutto IF – 2/6

- Quando un processo decisionale comporta più azioni, è possibile utilizzare, insieme alla clausola `if`, anche le seguenti clausole
  - `else`
    - Che caratterizza una **selezione a due vie**
  - `elseif`
    - Che caratterizza una **selezione "a cascata"**

Costrutto IF – 2/6

- **Selezione a Due Vie (Sintassi MATLAB)**

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



Costrutto IF – 2/6

- **Selezione a Due Vie (Sintassi MATLAB)**

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

- **Se** l'`espressione` è vera (1) **allora** le istruzioni del blocco 1 (`blocco_istruzioni_1`) sono eseguite
- **Altrimenti** (se l'`espressione` è falsa (0) allora) le istruzioni del blocco 2 (`blocco_istruzioni_2`) sono eseguite

Costrutto IF – 2/6

- **Selezione a Due Vie (Sintassi MATLAB)**

```
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 – 2/6

- **Selezione a Due Vie (Sintassi MATLAB)**

```
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 – 2/6

- **Selezione a Due Vie (Sintassi MATLAB)**

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

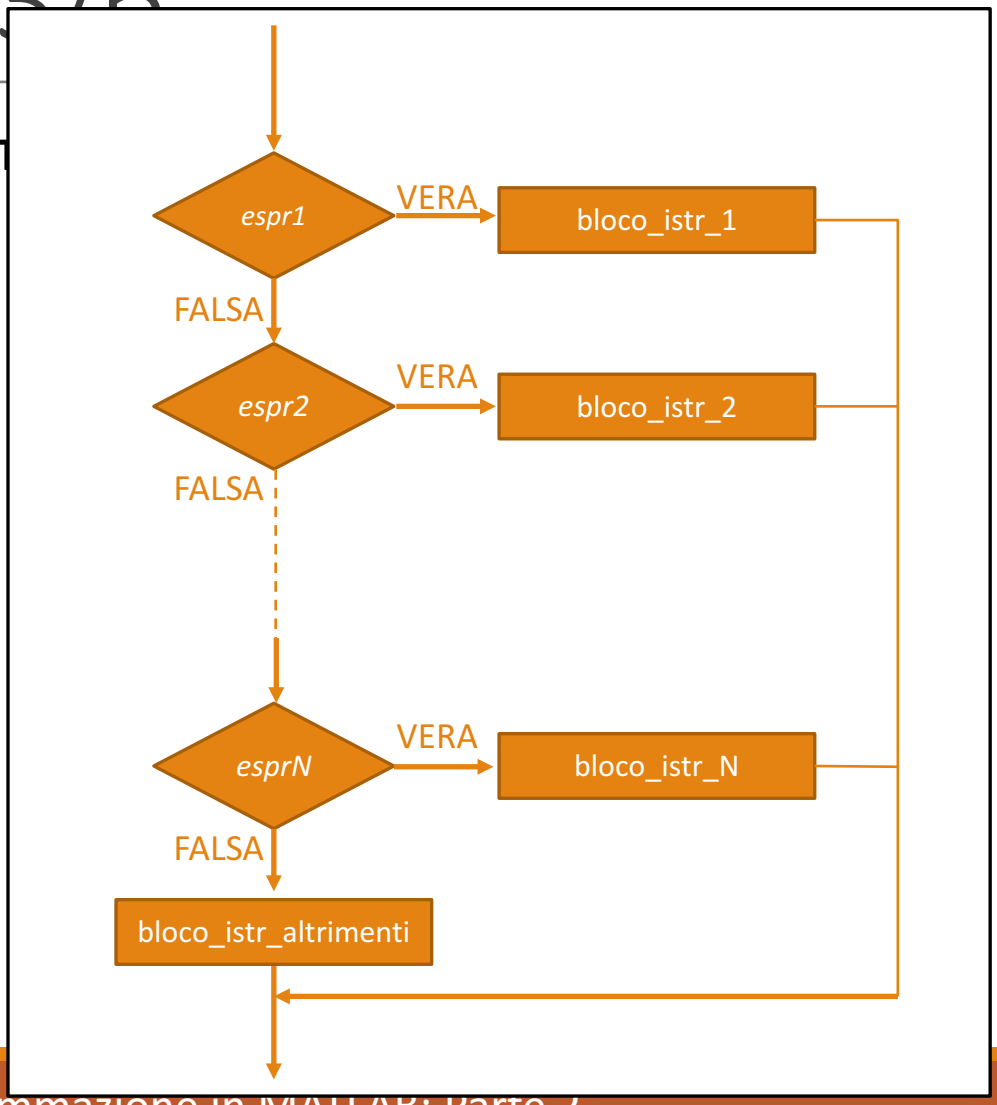
- ***Esempio 3***

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

Costrutto IF – 3/6

- **Selezione a «Cascata» (Sintassi MAT**

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



Costrutto IF – 3/6

- **Selezione a «Cascata» (Sintassi MATLAB)**

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

Costrutto IF – 3/6

- **Selezione a «Cascata»: Esempio 1**

```
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('Nessuno sconto! Spendi di più per averlo!!!')
else
    disp('Prezzo non corretto!')
end
```

Costrutto IF – 3/6

- **Selezione a «Cascata»: Esempio 1**

```
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('Nessu                                    lo!!')
else
    disp('Prezz
end
```

NOTA: Equivale a

$$599 \leq prezzo < 999$$

Costrutto IF – 3/6

- **Selezione a «Cascata»: Esempio 1**

```
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('Nessuno sconto! $
else
    disp('Prez
end
```

N.B.

- Le clausole **`else`** ed **`elseif`** possono essere omesse quando non necessario
- Tuttavia, se vengono utilizzate entrambe, la clausola **`else`** deve essere posta dopo **`elseif`**

Costrutto IF – 3/6

- **Selezione a «Cascata»: Esempio 1**

```
if prezzo >= 999
    perc_sconto = 35;
elseif prezzo >= 599 && prezz
    perc_sconto = 25;
elseif prezzo < 599
    perc_sconto = 19;
elseif prezzo <= 0.99
    perc_sconto = 0;
    disp('Nessuno sconto! Spendi di più per averlo!!')
else
    disp('Prezzo non corretto!')
end
```

N.B.

- La clausola **`elseif`** non richiede un'apposita clausola **`end`**

Costrutto IF – 4/6

- Le **strutture decisionali** possono essere **annidate**
  - Una struttura può essere inclusa all'interno di un'altra struttura che, a sua volta, può contenerne un'altra e così via…
- Il rientro verso destra di alcune righe del codice mette in evidenza i gruppi delle istruzioni associate a ciascuna clausola `end`

```
if espressione1
    blocco_istruzioni_1
    if espressione2
        blocco_istruzioni_2
    end
end
```

- Si noti che ogni clausola `if` è associata ad una clausola `end`

Costrutto IF – 5/6

- ***Esempio 4***

N.B. Ciascun `else` fa riferimento all'ultimo `if` che non è stato chiuso da una clausola `end`

```
cena = true;
fa_i_compiti = true;
sistema_stanza = false;

if cena && fa_i_compiti
    if ~sistema_stanza
        disp('Bravo/a ragazzo/a');
    else
        disp('Ragazzo/a eccezionale');
    end
else
    disp('Ragazzaccio/a');
end
```

```
color = input('Pensa a un animale ed inserisci il suo colore >> ', 's');
h = input('Inserisci la sua altezza in cm >> ');

if strcmp(color, 'verde')
    if (h < 20)
        disp('È una rana!');
    else
        disp('È un coccodrillo!');
    end
elseif strcmp(color, 'grigio')
    if h < 100
        disp('È un vombatide!');
    else
        disp('È un orso!');
    end
else
    disp('Non conosco questo animale!');
end
```

```
color = input('Pensa a un animale ed inserisci il suo colore >> ', 's');
h = input('Inserisci la sua altezza in cm >> ');

if strcmp(color, 'verde')
    if (h < 20)
        disp('È una rana!');
    else
        disp('È un coccodrillo!');
    end
elseif strcmp(color, 'grigio')
    if h < 100
        disp('È un vombatide!');
    else
        disp('È un orso!');
    end
else
    disp('Non conosco questo animale!');
end
```

```
color = input('Pensa a un animale ed inserisci il suo colore >> ', 's');
h = input('Inserisci la sua altezza in cm >> ');

if strcmp(color, 'verde')
    if (h < 20)
        disp('È una rana!');
    else
        disp('È un coccodrillo!'
    end
elseif strcmp(color, 'grigio')
    if h < 100
        disp('È un vombatide!');
    else
        disp('È un orso!');
    end
else
    disp('Non conosco questo animale!');
end
```

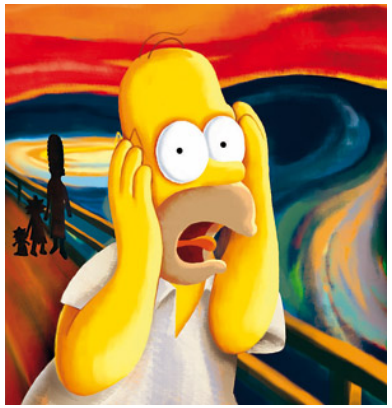
N.B. Di default la funzione `input` interpreta l'input come un numero, l'opzione (parametro) `'s'` permette di interpretare l'input come una stringa

```
color = input('Pensa a un animale ed inserisci il suo colore >> ', 's');
h = input('Inse                                altezza in cm >> ');

if strcmp(color, 'verde')
    if (h < 20)
        disp('È una rana!');
    else
        disp('È un coccodrillo!');
    end
elseif strcmp(color, 'grigio')
    if h < 100
        disp('È un vombatide!');
    else
        disp('È un orso!');
    end
else
    disp('Non conosco questo animale!');
end
```

help strcmp

```
color = input('Pensa a un animale ed inserisci il suo colore >> ', 's');
h = input('Inserisci la sua altezza in cm >> ');

if strcmp(color, 'verde')
if (h < 20)
disp('È una rana!');
else
disp('È un coccodrillo!');
end
elseif strcmp(color, 'grigio')
if h < 100
disp('È un vombatide!');
else
disp('È un orso!');
end
else
disp('Non conosco questo animale!');
end
```

N.B. Prestare sempre **molta attenzione** all'indentazione del codice, che risulterà altrimenti assai difficile da leggere e comprendere



Costrutto IF – 6/6

Selezione con IF annidati (Esempio Anno Bisestile)

Anno bisestile (Definizione Enciclopedia Treccani): "L'anno di 366 giorni, con il febbraio di 29 giorni. Nella riforma gregoriana è bisestile un anno ogni quattro, esclusi gli anni secolari il cui numero non sia divisibile per 400."

Costrutto IF – 6/6

Selezione con IF annidati (Esempio 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 – 6

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**

- **Selezione con *IF annidati***

Costrutto IF – 6

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 È BISESTILE***
- Fine (Se)

- **Selezione con *IF annidati***

Costrutto IF – 6

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**

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

- **Selezione con *IF annidati***

Costrutto IF – 6

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**

- **Selezione con *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 ***È BISESTILE***
    - Altrimenti
      - L'anno ***NON È BISESTILE***
    - Fine (Se)
  - Altrimenti
    - L'anno ***È BISESTILE***
  - Fine (Se)
- Altrimenti
  - L'anno ***NON È BISESTILE***
- Fine (Se)

Costrutto IF – 6

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 ***È BISESTILE***
    - Altrimenti
      - L'anno ***NON È BISESTILE***
    - Fine (Se)
  - Altrimenti
    - L'anno ***È BISESTILE***
  - Fine (Se)
- Altrimenti
  - L'anno ***NON È BISESTILE***
- Fine (Se)

Costrutto IF – 6

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 ***È BISESTILE***
    - Altrimenti
      - L'anno ***NON È BISEST***
    - Fine (Se)
  - Altrimenti
    - L'anno ***È BISESTILE***
  - Fine (Se)
- Altrimenti
  - L'anno ***NON È BISESTILE***
- Fine (Se)

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

- **Selezione con *IF annidati***

Costrutto IF – 6

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 ***È BISESTILE***
    - Altrimenti
      - L'anno ***NON È BISESTILE***
    - Fine (Se)
  - Altrimenti
    - L'anno ***È BISESTILE***
  - Fine (Se)
- Altrimenti
  - L'anno ***NON È 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)

- **Selezione con *IF annidati***

Costrutto IF – 6

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 ***È BISESTILE***
    - Altrimenti
      - L'anno ***NON È BISESTILE***
    - Fine (Se)
  - Altrimenti
    - L'anno ***È BISESTILE***
  - Fine (Se)
- Altrimenti
  - L'anno ***NON È 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 – 6

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**

- **Selezione con *IF annidati***

- 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 ***È BISESTILE***
        - Altrimenti
            - L'anno ***NON È BISESTILE***
        - Fine (Se)
    - Altrimenti
        - L'anno ***È BISESTILE***
    - Fine (Se)
- Altrimenti
    - L'anno ***NON È BISESTILE***
- Fine (Se)

Costrutto IF – 6

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**

- **Selezione con *IF annidati***

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

- ***Codice MATLAB (M-File Script anno_bisestile.m)***

```
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 – 6/6

- ***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

```
>> 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 – 1/4

- Il costrutto `switch-case` consente di selezionare un gruppo di istruzioni da eseguire, in base al valore di una `variabile` presa in input

```
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
```

- Tale costrutto risulta essere utile per le enumerazioni ma poco appropriato per gli intervalli

Costrutto SWITCH-CASE – 2/4

- ***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 – 3/4

- ***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
```

Costrutto SWITCH-CASE – 4/4

- Il costrutto `switch-case` consente di selezionare un gruppo di istruzioni da eseguire, in base al valore di una `variabile` presa in input

```
switch variabile
    case valore1
        blocco_istruz
    case valore2
        blocco_istruz
     .
     .
     .
    case valoreN
        blocco_istruzioni_valore_N
    otherwise

blocco_istruzioni_otherwise
end
```

- N.B. Ogni `valore` di `case` deve essere scritto su una singola riga del programma

- Tale costrutto risulta essere utile per le enumerazioni ma poco appropriato per gli intervalli

Costrutto SWITCH-CASE – 4/4

- Il costrutto `switch-case` consente di selezionare un gruppo di istruzioni da eseguire, in base al valore di una `variabile` presa in input

```
switch variabile
    case valore1
        blocco_istruzi
    case valore2
        blocco_istr
     .
     .
     .
    case valoreN
        blocco_istruzi
    otherwise

blocco_istruzioni_othe
end
```

- La `variabile` di input viene confrontata con il `valore` che segue ciascuna clausola `case`
- Se esiste una clausola `case` seguita da un `valore` identico a quello assunto dalla `variabile` presa in input, MATLAB esegue il `blocco di istruzioni` relativo a tale clausola
  - Poi prosegue l'elaborazione con le istruzioni che si trovano dopo la relativa clausola `end`

- Tale costrutto risulta essere utile per le
per gli intervalli

Costrutto SWITCH-CASE – 4/4

- Il costrutto `switch-case` consente di selezionare un gruppo di istruzioni da eseguire, in base al valore di una `variabile` presa in input

```
switch variabile
    case valore1
        blocco_istruz
    case valore2
        blocco_istruz
     .
     .
     .
    case valoreN
        blocco_istruzioni_valore_N
    otherwise

blocco_istruzioni_otherwise
end
```

- N.B. vengono eseguite soltanto le istruzioni corrispondenti al primo `valore` di `case` equivalente alla `variabile` di input

- Tale costrutto risulta essere utile per le enumerazioni ma poco appropriato per gli intervalli

Costrutto SWITCH-CASE – 4/4

- Il costrutto `switch-case` consente di selezionare un gruppo di istruzioni da eseguire, in ba

```
sw
    case va
        bloc  istruzioni_valore_N
    otherwise

    blocco_istruzioni_otherwise
    end
```

- Se nessun `valore` di `case` è equivalente alla `variabile` di input, vengono eseguite le istruzioni che seguono la clausola `otherwise`
  - N.B. questa clausola è facoltativa: se viene omessa e nessun `valore` di `case` è equivalente a quello della `variabile` di input, l'elaborazione continua con le istruzioni che seguono la clausola `end`

- Tale costrutto risulta essere utile per le enumerazioni ma poco appropriato per gli intervalli

Riferimenti

- Capitolo 4 (MATLAB, Un'introduzione per gli ingegneri)
  - Paragrafi 1, 2, 3 e 4