

FONDAMENTI DI INFORMATICA

QUINTA LEZIONE

Prof. Alfredo Accattatis

(accattatis@ing.uniroma2.it)

Tutor:

Prof. Vittorio Emanuele Calafiore

(vcalafio@gmail.com)

Esempio di funzione(2)

Parola riservata
function

Argomenti d'uscita

Due argom. ingresso

function outvec = createvec(mymin, mymax)

% createvec crea un vettore di valori che variano tra un
% minimo ed un massimo
% restituisce il vettore

Nome della funzione

Assegnazione

% se il minimo non è minore del massimo, scambia i valori

if mymin > mymax
 temp = mymin;
 mymin = mymax;
 mymax = temp;

end

outvec = mymin:mymax;

end

- Salviamo la funzione in un M-file (createvec.m, stesso nome della funzione)
- Chiamiamo la funzione

```
>> createvec(7, 3)
```

```
ans =
```

```
3 4 5 6 7
```

Concetto di ITERAZIONE

- **L'Iterazione** permette la ripetizione controllata di un blocco di codice (“loop”)
- Istruzioni di controllo all'inizio del blocco di codice definisce i limiti (ed il modo) della ripetizione :
 - L'istruzione di loop “**for**” è definita per ripetere un blocco di codice un **numero prefissato di volte**
 - L'istruzione di loop “**while**” è più flessibile : consente di ripetere il blocco di codice un numero di volte variabile (dipende dalla “condizione”)

for loop

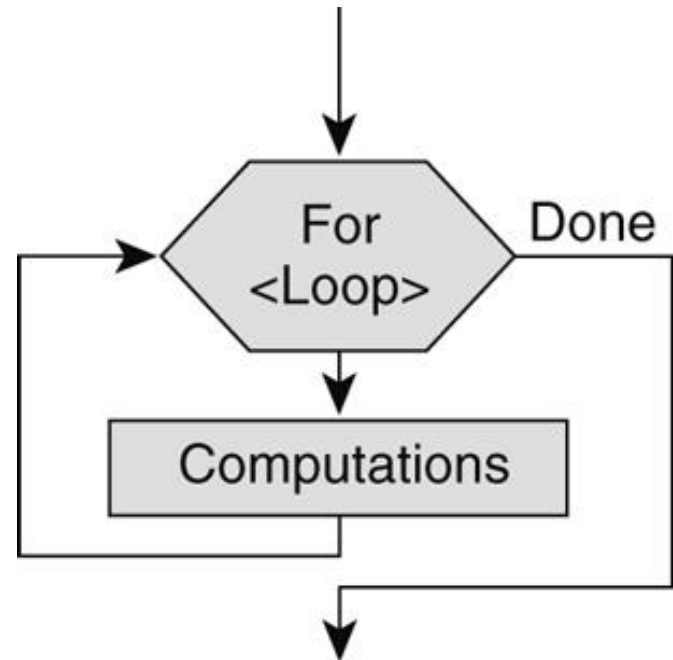
Lo schema base del for loop è:

```
for loopvar = vector  
    <blocco di codice>  
end
```

Più in generale:

```
for variable = expr  
    <blocco di codice>  
end
```

Il “for loop” automaticamente setta il valore della variabile di controllo del loop ad ogni giro ed esegue il blocco di codice con quel valore

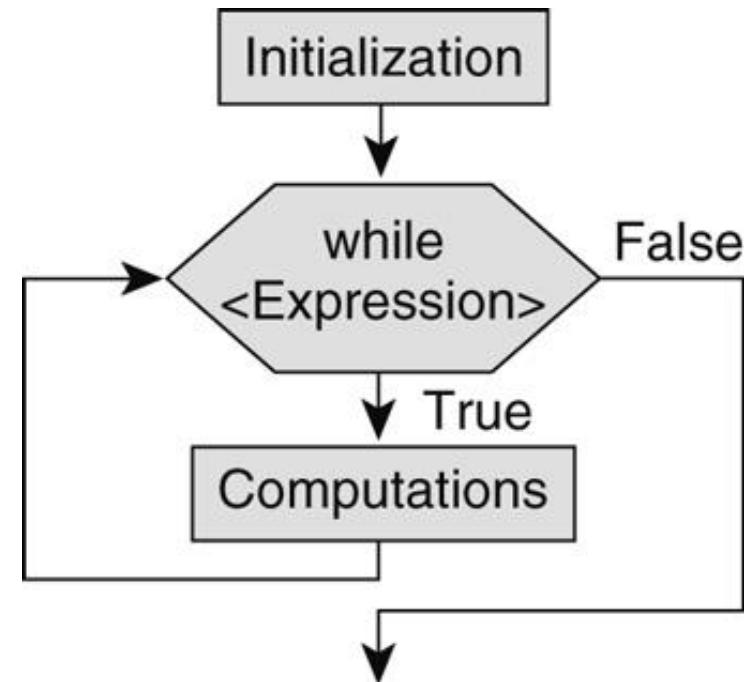


while loop

Il blocco di codice è ripetuto fino a che l'espressione logica rimane vera

In generale:

```
<initialization>  
while <logical expression>  
    <code block>  
    % per far terminare il ciclo  
    % bisogna modificare l'espressione logica  
end
```



Esempi di loop

```
% implementa la funzione zeros(1,N)
N=10;
for i=1:N          % specifichiamo il range dei valori
    v(i)=0;        % usando l'operatore colonna
end

% setta gli elementi dispari di v al valore 1
for i=1:2:N
    v(i)=1;
end

%lo stesso codice di cui sopra può essere scritto come:
for i=[1,3,5,7,9]
    v(i)=1;
end
```

Esempi di loop con while

```
% implementa la funzione zeros(1,N)
N=10;
for i=1:N           % specifichiamo il range dei valori
    v(i)=0;         % usando l'operatore colonna
end
```

```
% implementa la funzione zeros(1,N)
N=10;
i = 0;
while i<=N           % specifichiamo il range dei valori
    v(i)=0;          % scorrendo direttamente i valori
    i = i + 1;
end
```

La funzione PLOT

- Ci consente di disegnare grafici a due e tre dimensioni
- E' molto semplice, la sua forma più elementare richiede solo due parametri (X e Y)
- Esempio: `plot(x, y)`
- Se assumiamo: $x = 0:\pi/100:2*\pi$;
- E: $y = \sin(x)$
- Allora
- `plot(x,y)`

La funzione **find**

- Una funzione molto utile, dato $B = [1 \ 0 \ 5 \ 0 \ 0]$

```
>> C = find(B)
```

```
ans =
```

```
    [1  3]
```

- La funzione `find(...)` prende in “ingresso” un array (vettore, matrice, per esempio logico) e restituisce un vettore “lineare” di indici che identificano gli elementi “veri” (true) del vettore di ingresso (in generale gli elementi >0)

Funzioni di libreria

Tutte le funzioni MATLAB accettano vettori di numeri piuttosto che singoli valori e restituiscono un vettore della medesima lunghezza.

Funzioni Speciali:

- **sum(v)** e **mean(v)** prende in ingresso un vettore e ritorna uno scalare

- **min(v)** e **max(v)** ritornano il minimo od il massimo valore di un vettore, oltre alla posizione

- **round(v)** , **ceil(v)** , **floor(v)** , e **fix(v)** rimuovono la parte frazionaria secondo diverse strategie: arrotondamento, arrotondamento verso l'alto, verso il basso, verso lo zero

Iniziamo a codificare algoritmi

- Useremo i vettori, poi delle matrici
- Definiremo algoritmi per risolvere problemi
- Li rappresenteremo con diagrammi di flusso
- Li tradurremo con le istruzioni da riga di comando (le uniche che per ora conosciamo)
- Interpretato : file .m, eseguito una linea alla volta
- Compilato: dal file .m tradotto una volta per tutte generando un file .exe

Manipolazione di vettori: esempi

1. Estrarre gli elementi di v con indice dispari
2. Estrarre gli elementi di v con indice pari
3. **Concatenare i due vettori** in un nuovo vettore (dispari, pari)
4. Trovare ed estrarre gli elementi di v maggiori di un valore **$y_{\min}$**
5. Estrarre gli elementi di v compresi tra **$y_{\min}$** e **$y_{\max}$**
6. Calcolare la **somma** e la **media** degli elementi del vettore

Estrarre gli elementi di un vettore v con indice dispari e memorizzarli in w

1. Leggiamo v e la sua dimensione N
2. se $N==1$
 1. estrai il solo elemento dispari $v(1)$
3. altrimenti *%almeno 2 elementi*
 1. inizializza i contatori $i=0$; $j=0$;
 2. $i=i+1$;
 3. se $\text{mod}(i,2)==0$ allora i è un elemento pari, vai al passo 3.5
 4. altrimenti *%elemento con indice dispari*
 1. $j=j+1$;
 2. $w(j)=v(i)$;
 5. se $i<N$ vai a passo 3.2 altrimenti vai al passo 4 (FINE)
4. FINE

$\text{MOD}(x,y)$ is $x - n.*y$ where
 $n = \text{floor}(x./y)$ if $y \neq 0$.

Ad es: $y=2$

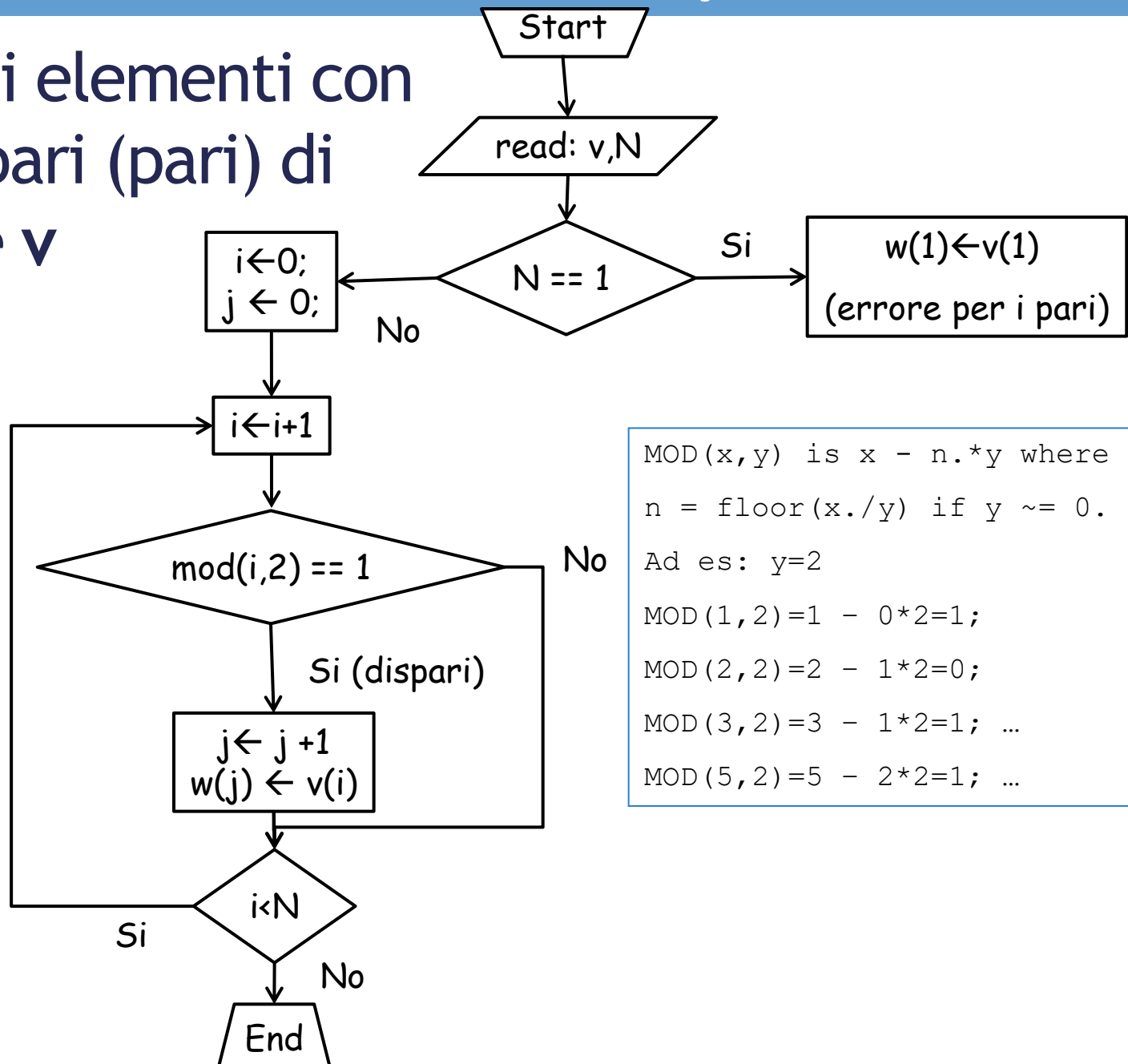
$\text{MOD}(1,2)=1 - 0*2=1$;

$\text{MOD}(2,2)=2 - 1*2=0$;

$\text{MOD}(3,2)=3 - 1*2=1$; ...

$\text{MOD}(5,2)=5 - 2*2=1$; ...

Estrarre gli elementi con indice dispari (pari) di un vettore v



Soluzione con Matlab (linea di comando)

```
>> v=linspace(1,3,5)
```

```
v =
```

```
1.0000    1.5000    2.0000    2.5000    3.0000
```

```
>> wd=v(1:2:end)
```

```
wd =
```

```
1         2         3
```

```
>> wp=v(2:2:end)
```

```
wp =
```

```
1.5000    2.5000
```

```

% estrae gli elementi con indice pari
v = [ 1 4 6 8 3 5 9 ];
N = length( v );
w = zeros( 1, N );
continua = true; % variabile per interruzione ciclo while
i = 0; % contatore generale
j = 0; % contatore dei dispari

```

```

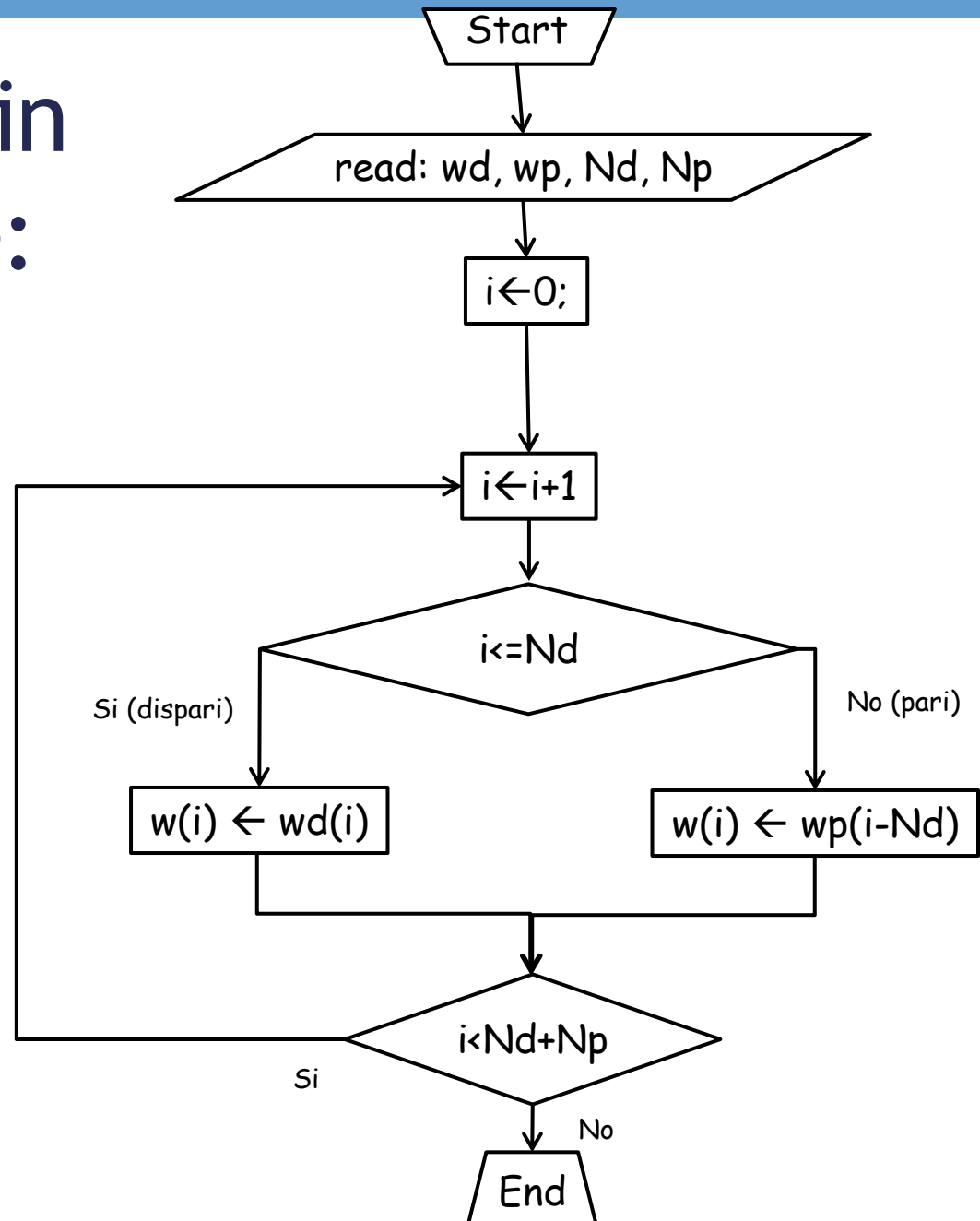
if (N == 1)
    w(1) = v(1);
else
    while continua == true
        i = i + 1;
        if ( mod( i,2 ) == 1 ) % verifica se dispari
            j = j + 1;
            w( j ) = v( i );
        end
        if i < N % verifica se algoritmo finito
            continua = true;
        else
            continua = false;
        end
    end
end
disp(w)

```

Homework

- Implementate lo stesso algoritmo con uno script che usi:
 - Istruzioni condizionali
 - Loop (for, while già usato)
- In fase successiva, scrivere una function

Concateniamoli in
un nuovo vettore:
(dispari, pari)



Soluzione Matlab linea di comando

```
>> w=[wd, wp]
```

```
w =
```

```
    1.0000    2.0000    3.0000    1.5000    2.5000
```

```
>> w=[wd; wp]
```

```
Error using vertcat Dimensions of matrices being  
concatenated are not consistent.
```

```
>> N=min(length(wd),length(wp))
```

```
N =
```

```
    2
```

```
>> w=[wd(1:N); wp(1:N)]
```

```
w =
```

```
    1.0000    2.0000
```

```
    1.5000    2.5000
```

Homework

- Implementate lo stesso algoritmo con uno script che usi:
 - Istruzioni condizionali
 - Loop (for e while)
- In fase successiva, scrivere una function

Array: cenni

Terminologia informatica: **ARRAY** è un insieme di dati omogenei organizzato in maniera che ciascun elemento (dato) è identificato univocamente.

Es: contenitore con tanti cassette.

Scalari, vettori e matrici sono delle tipologie particolari di array:

Scalare: array di un unico elemento (1×1)

Vettore: array monodimensionale ($m \times 1$ o $1 \times n$)

Matrice: array bidimensionale ($m \times n$)

Concatenazione

- MATLAB consente la costruzione di un(a) nuovo(a) vettore(matrice), concatenando più vettori(matrici):

- $A = [B \ C \ D \ \dots \ X \ Y \ Z]$

I singoli elementi possono essere vettori (matrici) definiti come costanti o variabili, e la lunghezza di A sarà la somma delle lunghezze dei singoli vettori.

- $A = [1 \ 2 \ 3 \ 42]$

...è un caso particolare in cui le singole componenti sono vettori ad una sola componente (scalari).

MATRICI

Operazioni basilari:

- Creazione
- Estrazione/inserimento dati ed elementi
- Operazioni aritmetiche e logiche
- Riduzione/trasformazione di matrice

Creazione di una matrice

- Inserimento manuale dei valori

```
>> A=[2 4 5 6; 3 5 6 7]
```

```
A =
```

2	4	5	6
3	5	6	7

- Funzioni predefinite

```
zeros(r, c)
```

```
ones(r, c)
```

```
rand(r, c)
```

```
eye(r, c)
```

Matrici: utilizzo degli indici

- Sintassi:

- $A(r, c)$ estrae l'elemento nella posizione specificata.

```
>> A(2, 3)
ans =
     6
```

$$A = \begin{pmatrix} 2 & 4 & 5 & 6 \\ 3 & 5 & 6 & 7 \end{pmatrix}$$

- $A(r, c) =$ sostituisce il valore dell'elemento nella posizione specificata.

```
>> A(2, 3) = 4
A =
     2     4     5     6
     3     5     4     7
```

- L'indice può essere un valore intero oppure logico

```
>> B=[true false false false; false false true false];
>> A(B)
ans =
     2
     6
```

Indici numerici

- E' possibile estrarre una porzione di matrice

```
>> A(1, 2:4)
ans =
    4    5    6
```

$$A = \begin{pmatrix} 2 & 4 & 5 & 6 \\ 3 & 5 & 6 & 7 \end{pmatrix}$$

- Utilizzando ':' si estraggono tutte le righe o colonne

```
>> A(:, 3)
ans =
    5
    6
```

- **Linear indexing:** si utilizza un solo indice - MATLAB memorizza gli elementi della matrice colonna per colonna (*columnwise*)

```
>> A(3)
ans =
    4
```

Indici logici

```
>> B=[true false false false;  
      false false true false];
```

```
>> A(B)
```

```
ans =
```

```
2
```

```
6
```

$$A = \begin{pmatrix} 2 & 4 & 5 & 6 \\ 3 & 5 & 6 & 7 \end{pmatrix}$$

- Un elemento della matrice è selezionato mediante il valore TRUE nella relativa posizione della matrice booleana
- La matrice booleana deve avere i valori TRUE o FALSE: se inizializzata ad 1 o 0 è una matrice di double.

Operazioni sulle matrici

- Scalari :

+ - * /

- Array di dati (membro a membro):

+ - .* ./ .^

- Moltiplicazione matriciale:

*

- Concatenazione:

horzcat() vertcat()

- Trasposizione:

'

- Relazionali e logiche:

< > == | &

- Funzioni built-in:

sin(), isequal(), ...

Prodotto tra matrici

$$[C]_{m \times p} = [A]_{m \times n} * [B]_{n \times p}$$

$$c_{ij} = \sum_{k=1}^n a_{ik} b_{kj}$$

- Ad esempio

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

```
>> B=[1 2 3 1; 4 5 1 2; 0 2 3 0];
```

```
>> C = A*B
```

```
C =
```

```
    35    46    17    19
```

```
     9    22    20     5
```

```
>> D = B*A
```

Error using *

Inner matrix dimensions must agree.

Concatenazione

- La concatenazione può essere sia orizzontale che verticale
- Orizzontale** $[A \ B \ C]$ oppure `horzcat(A,B,C)`
 - le matrici operando hanno lo stesso numero di righe

$$\left(\begin{pmatrix} 2 & 4 & 5 & 6 \\ 3 & 5 & 6 & 7 \end{pmatrix} \right) \left(\begin{pmatrix} 2 & 4 & 5 & 6 \\ 3 & 5 & 6 & 7 \end{pmatrix} \right) \left(\begin{pmatrix} 2 & 4 & 5 & 6 \\ 3 & 5 & 6 & 7 \end{pmatrix} \right)$$

- Verticale** $[A; B; C]$ oppure `vertcat(A,B,C)`
 - le matrici operando hanno lo stesso numero di colonne

$$\left(\begin{pmatrix} 2 & 4 & 5 & 6 \\ 3 & 5 & 6 & 7 \end{pmatrix} \right) \left(\begin{pmatrix} 2 & 4 & 5 & 6 \\ 3 & 5 & 6 & 7 \end{pmatrix} \right) \left(\begin{pmatrix} 2 & 4 & 5 & 6 \\ 3 & 5 & 6 & 7 \end{pmatrix} \right)$$

Matrice trasposta

$$A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix} = a_{ij} \quad i = 1, \dots, m; \quad j = 1, \dots, n$$

$$A = \begin{pmatrix} 2 & 4 & 5 & 6 \\ 3 & 5 & 6 & 7 \end{pmatrix}$$

```
>> A'
ans =
```

```

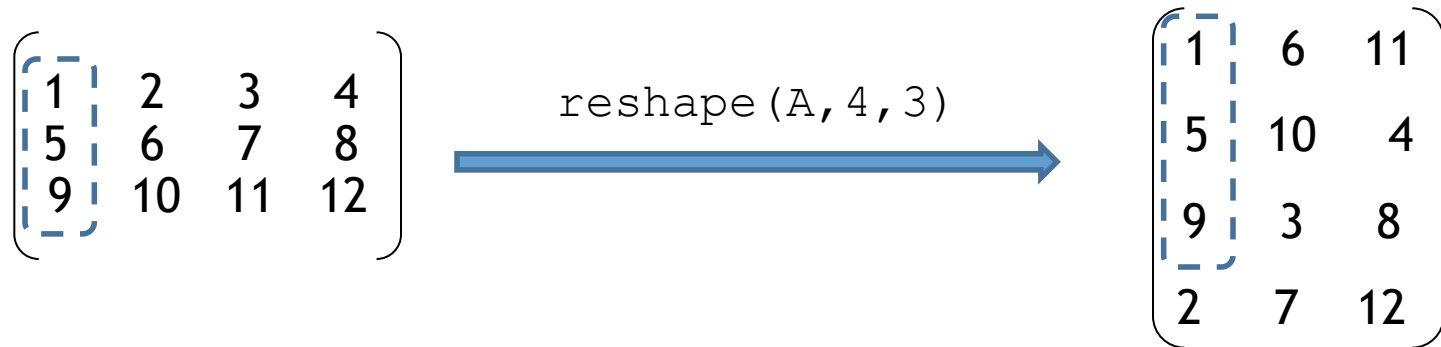
2      3
4      5
5      6
6      7
```

 $A'_{(n \times m)} =$

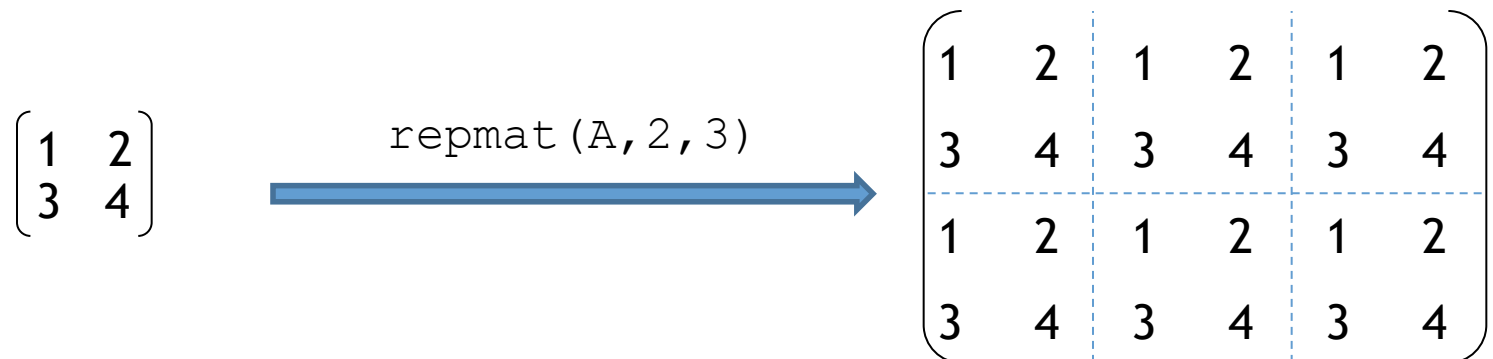
$$\begin{bmatrix} a_{11} & a_{21} & \cdots & a_{m1} \\ a_{12} & a_{22} & \cdots & a_{m2} \\ \vdots & & \ddots & \\ a_{1n} & a_{2n} & \cdots & a_{mn} \end{bmatrix}$$

Modifica delle dimensioni

- `reshape(A, p, q)` : riorganizza **columnwise** gli elementi di una matrice ($n \times m$) in una nuova matrice ($p \times q$), dove $n*m = p*q$



- `repmat(A, s, t)` : crea la matrice contenente $s*t$ copie di A



Proprietà di una matrice

- `size()`: [r, c]
 - `size(A) = [2 4]`
- `length()`: maggiore tra r e c
 - `length(A) = 4`
- `numel()`: r X c
 - `numel(A) = 8`
- `sum()`: in una matrice somma gli elementi delle colonne
 - `sum(A) = [5 9 11 13]`
 - `sum(sum(A)) = 38`
- `det()`: determinante - solo matrici quadrate (n X n)
- `find()`: fornisce gli indici lineari degli elementi non nulli
 - `find(A>3 & A<6) = [3; 4; 5]`

$$A = \begin{pmatrix} 2 & 4 & 5 & 6 \\ 3 & 5 & 6 & 7 \end{pmatrix}$$

Esercizi (1)

Creare una matrice M (5 X 6) di numeri casuali con valore v dove $20 \leq v \leq 100$.

Sostituire gli elementi di valore ($v > 40$) e ($v < 50$), con il valore nullo.

Esercizi (2)

- `>> M = 20 + 80*rand(5,6)`
- `M =`

63.7772	40.3426	35.7276	86.4663	80.5760	24.3160
31.0900	85.1428	40.0867	66.8211	80.2983	62.4638
31.9435	39.4820	69.2836	63.9779	50.4357	82.3334
40.6007	94.3411	57.8631	93.3755	65.4257	94.7209
87.2574	47.9987	48.1328	42.8671	26.0683	30.3925
- `>> pos = find(M>40 & M<50)`
- `pos =`

4
6
10
12
15
20
- `>> M(pos) = 0`
- `M =`

63.7772	0	35.7276	86.4663	80.5760	24.3160
31.0900	85.1428	0	66.8211	80.2983	62.4638
31.9435	39.4820	69.2836	63.9779	50.4357	82.3334
0	94.3411	57.8631	93.3755	65.4257	94.7209
87.2574	0	0	0	26.0683	30.3925

Operazioni sulle matrici

- Scalari :

+ - * /

- Array di dati (membro a membro):

+ - .* ./ .^

- Moltiplicazione matriciale:

*

- Concatenazione:

horzcat() vertcat()

- Trasposizione:

'

- Relazionali e logiche:

< > == | &

- Funzioni built-in:

sin(), isequal(), ...

Ingresso - Uscita (Input / output)

- Da dove provengono i dati d'ingresso?
 1. **File esterni**
 2. **Utente**: dobbiamo invitare l'utente a digitare i dati (prompting), i.e., dire all'utente quali dati immettere
- Uno dei due è definito come il **dispositivo di ingresso di "default"**

- Dove va a finire l'Output?
 1. **File esterni**
 2. **Schermo**
- Uno dei due è il **dispositivo di uscita di "default"**

Ingresso - Uscita

- Come è possibile leggere il valore del raggio da una sorgente esterna?
- Come presentare i dati di uscita in maniera più efficace ed efficiente possibile?
- Si usano le *istruzioni* di **input/output (I/O)**

Istruzioni di Input

- La più semplice istruzione per l'ingresso dei dati in MATLAB è: **input**
- Da usare in una espressione di assegnazione

```
>> rad = input('Digitare il raggio: ')
Digitare il raggio: 5
rad =
5
```
- Per poter usare un carattere (o stringa) usare **'c'** o **'s'** come secondo argomento della funzione

```
>> mystr = input('Digita una stringa: ', 's')
```

 - Restituisce una stringa vuota se l'utente digita solo spazi o tabs
 - Se si usa un tipo NON previsto (esempio numero al posto di un carattere o viceversa) viene segnalato un messaggio di errore

Istruzioni per l'Output

- Le istruzioni per l'Output dei dati consentono di visualizzare a video stringhe e/o risultati di espressioni e ne consentono la **formattazione** (i.e., personalizzare le modalità di visualizzazione)
- La funzione `disp`, per esempio, permette di visualizzare sul display (il video) il risultato di una espressione oppure una stringa, ma non ne permette la formattazione

```
>> disp('Hello world! ')\nHello world!
```

- Invece la funzione `fprintf` consente di stampare (a video, e non solo) **con formattazione**

```
>> fprintf('il valore è %d, come previsto!\\n', 4^3)\nIl valore è 64, come previsto!\n>>
```

Funzione `fprintf`

Parametri d'ingresso della funzione `fprintf`:

- **Stringa di formattazione**: qualsiasi testo da stampare e le informazioni di formattazione per l'espressione da stampare
- **Place holder**: conversion character (d,f,c,s)
 - `%d` per valori interi
 - `%f` per valori reali
 - `%c` per singoli caratteri
 - `%s` per stringhe
- Carattere “a capo” (newline) `'\n'`
- Un **campo larghezza** può essere incluso nel place holder per specificare quanti caratteri devono essere usati

```
>> fprintf('Il valore di pi-greco è:%.2f (%f)\n', pi, pi)
Il valore di pi-greco è: 3.14 (3.141593)
```

Funzione `fprintf` : approfondire place holder e conversion character

Usare help:

`help fprintf`

Poi cliccare su:

`formatSpec`

Stampare vettori e matrici

- `fprintf` per un vettore:
 - Se un carattere di conversione ed il carattere “newline” sono in una stringa di formattazione, il vettore sarà stampato come colonna
 - Senza il carattere newline, sarà stampato come riga
- `fprintf` per una matrice
 - Specificando solo un carattere di conversione ed il carattere “newline”, gli elementi della matrice saranno stampati in una colonna
 - Senza fare uso di “newline” gli elementi verranno stampati in una riga
- Per stampare vettori e matrici, `disp` è più facile da usare rispetto a `fprintf` !

Esercizio su fprintf

- Provate a stampare una matrice con fprintf