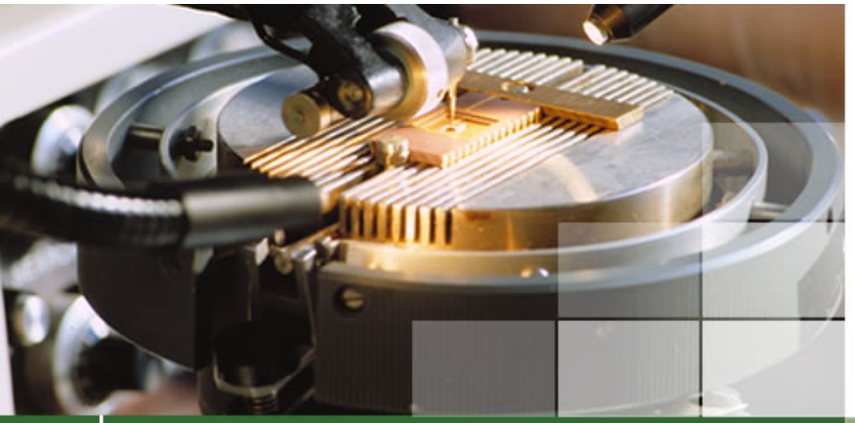


POLITECNICO DI MILANO

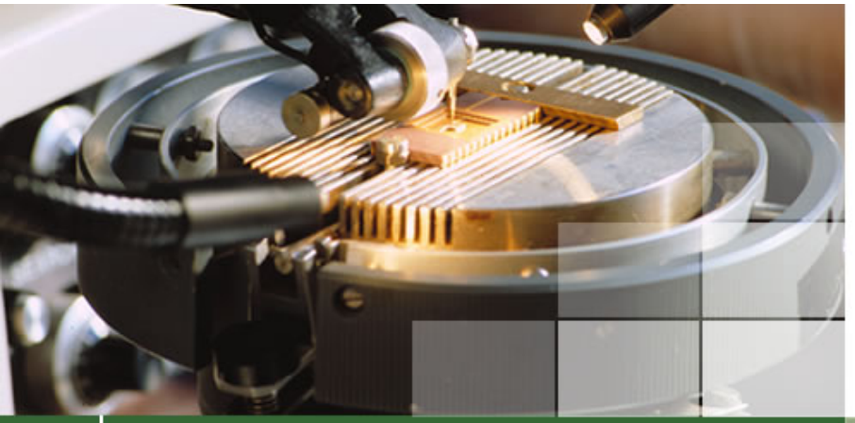
Dipartimento di
Elettronica e Informazione

MATLAB, from zero to hero

Matteo Ferroni

matteo.ferroni@polimi.it

22/12/2015



POLITECNICO DI MILANO

Dipartimento di
Elettronica e Informazione

MATLAB, from zero to ~~hero~~ structs and functions

Matteo Ferroni

matteo.ferroni@polimi.it

22/12/2015

MATLAB, from zero to hero

MATLAB, from zero to hero

zero to hero

the definition is split in to two parts
zero: being a loser, someone with no aspirations or goals
hero: being amazing person who does what they please and makes people want to be them; the ultimate in someone being an all around great person

Wow that boy went from zero to hero in a matter of minutes!

MATLAB, from zero to ~~hero~~ structs and functions

zero to hero

the definition is split in to two parts
zero: being a loser, someone with no aspirations or goals
hero: being amazing person who does what they please and makes people want to be them; the ultimate in someone being an all around great person

Wow that boy went from zero to hero in a matter of minutes!


You must *feel* the force around you.


You must *feel* the force around you.

Agenda

(20’) Es1 - Vettori e funzioni base

(20’) Es1.2 - Vettori, vettori everywhere!

(10’) Es2 - Funzioni e stringhe

(20’) Es3 - Files, matrici e funzioni

(20’) Es4 - Maschere di bit (ordinamento v1.0)

(20’) Es5 - Maschere di bit (ordinamento v2.0)

(20’) Es6 - Struct (rilievi altimetrici)

(20’) Es7 - Struct (film)

Esercizio: Vettori e funzioni base

Esercizio: Vettori e funzioni base

Scrivere un programma che permetta all'utente di inserire un vettore di numeri interi. Dopo aver verificato che l'array inserito sia numerico, effettuare i seguenti controlli:

Esercizio: Vettori e funzioni base

Scrivere un programma che permetta all'utente di inserire un vettore di numeri interi. Dopo aver verificato che l'array inserito sia numerico, effettuare i seguenti controlli:

- Verificare se tutti i numeri sono positivi

Esercizio: Vettori e funzioni base

Scrivere un programma che permetta all'utente di inserire un vettore di numeri interi. Dopo aver verificato che l'array inserito sia numerico, effettuare i seguenti controlli:

- Verificare se tutti i numeri sono positivi
- Verificare se esiste un numero negativo

Esercizio: Vettori e funzioni base

Scrivere un programma che permetta all'utente di inserire un vettore di numeri interi. Dopo aver verificato che l'array inserito sia numerico, effettuare i seguenti controlli:

- Verificare se tutti i numeri sono positivi
- Verificare se esiste un numero negativo
- Applicare la radice quadrata a tutti i valori: ci sono dei valori complessi?

Esercizio: Vettori e funzioni base

Scrivere un programma che permetta all'utente di inserire un vettore di numeri interi. Dopo aver verificato che l'array inserito sia numerico, effettuare i seguenti controlli:

- Verificare se tutti i numeri sono positivi
- Verificare se esiste un numero negativo
- Applicare la radice quadrata a tutti i valori: ci sono dei valori complessi?
- Verificare se tutti i numeri sono pari e trovarne le posizioni

Esercizio: Vettori e funzioni base

Scrivere un programma che permetta all'utente di inserire un vettore di numeri interi. Dopo aver verificato che l'array inserito sia numerico, effettuare i seguenti controlli:

- Verificare se tutti i numeri sono positivi
- Verificare se esiste un numero negativo
- Applicare la radice quadrata a tutti i valori: ci sono dei valori complessi?
- Verificare se tutti i numeri sono pari e trovarne le posizioni
- Verificare se esiste un numero dispari

Esercizio: Vettori e funzioni base

Scrivere un programma che permetta all'utente di inserire un vettore di numeri interi. Dopo aver verificato che l'array inserito sia numerico, effettuare i seguenti controlli:

- Verificare se tutti i numeri sono positivi
- Verificare se esiste un numero negativo
- Applicare la radice quadrata a tutti i valori: ci sono dei valori complessi?
- Verificare se tutti i numeri sono pari e trovarne le posizioni
- Verificare se esiste un numero dispari
- Contare i numeri dispari se esistono e dire in che posizione sono

Esercizio: Vettori e funzioni base

```
% Scrivere un programma che permetta all'utente di
% inserire un vettore di numeri interi.


% Dopo aver verificato che l'array inserito sia numerico,
%effettuare i seguenti controlli:
```

Esercizio: Vettori e funzioni base

```
% Scrivere un programma che permetta all'utente di
% inserire un vettore di numeri interi.
vett = input('Inserisci un vettore: ');

% Dopo aver verificato che l'array inserito sia numerico,
%effettuare i seguenti controlli:
```

Esercizio: Vettori e funzioni base

```
% Scrivere un programma che permetta all'utente di
% inserire un vettore di numeri interi.
vett = input('Inserisci un vettore: ');

% Dopo aver verificato che l'array inserito sia numerico,
%effettuare i seguenti controlli:
if isnumeric(vett)

    ...

else
    disp('Devi inserire un array numerico');
end
```

Esercizio: Vettori e funzioni base

```
% Scrivere un programma che permetta all'utente di
% inserire un vettore di numeri interi.
vett = input('Inserisci un vettore: ');

% Dopo aver verificato che l'array inserito sia numerico,
%effettuare i seguenti controlli:
if isnumeric(vett)
    ...
else
    disp('Devi inserire un array numerico');
end
```

Esercizio: Vettori e funzioni base

```
% Verificare se tutti i numeri sono positivi
```

Esercizio: Vettori e funzioni base

```
% Verificare se tutti i numeri sono positivi
if all(vett > 0)
    disp('tutti i numeri sono positivi');
else
    disp('non tutti i numeri sono positivi');
end
```

```
% Verificare se esiste un numero negativo
```

Esercizio: Vettori e funzioni base

```
% Verificare se tutti i numeri sono positivi
if all(vett > 0)
    disp('tutti i numeri sono positivi');
else
    disp('non tutti i numeri sono positivi');
end
```

```
% Verificare se esiste un numero negativo
if any(vett <0)
    disp('esiste un numero negativo');
else
    disp('non esiste alcun numero negativo');
end
```

Esercizio: Vettori e funzioni base

```
% Applicare la radice quadrata a tutti i valori:
% ci sono dei valori complessi?
```

Esercizio: Vettori e funzioni base

```
% Applicare la radice quadrata a tutti i valori:
% ci sono dei valori complessi?
sqrt_vett = sqrt(vett)
```

Esercizio: Vettori e funzioni base

```
% Applicare la radice quadrata a tutti i valori:
% ci sono dei valori complessi?
sqrt_vett = sqrt(vett)
if isreal(sqrt_vett)
    disp('non esistono valori complessi');
else
    disp('esistono valori complessi');
end
```

```
% Verificare se tutti i numeri sono pari e
% trovarne le posizioni
```

Esercizio: Vettori e funzioni base

```
% Applicare la radice quadrata a tutti i valori:
% ci sono dei valori complessi?
sqrt_vett = sqrt(vett)
if isreal(sqrt_vett)
    disp('non esistono valori complessi');
else
    disp('esistono valori complessi');
end
```

```
% Verificare se tutti i numeri sono pari e
% trovarne le posizioni
vett_mod = mod(vett,2);
```

Esercizio: Vettori e funzioni base

```
% Applicare la radice quadrata a tutti i valori:
% ci sono dei valori complessi?
sqrt_vett = sqrt(vett)
if isreal(sqrt_vett)
    disp('non esistono valori complessi');
else
    disp('esistono valori complessi');
end
```

```
% Verificare se tutti i numeri sono pari e
% trovarne le posizioni
vett_mod = mod(vett,2);
if(all(vett_mod==0))
    disp('tutti i numeri sono pari');
else
    disp('non tutti i numeri sono pari');
end
```

Esercizio: Vettori e funzioni base

```
% Applicare la radice quadrata a tutti i valori:
% ci sono dei valori complessi?
sqrt_vett = sqrt(vett)
if isreal(sqrt_vett)
    disp('non esistono valori complessi');
else
    disp('esistono valori complessi');
end
```

```
% Verificare se tutti i numeri sono pari e
% trovarne le posizioni
vett_mod = mod(vett,2);
if(all(vett_mod==0))
    disp('tutti i numeri sono pari');
else
    disp('non tutti i numeri sono pari');
end
pos_pari = find(1-vett_mod);
disp('I numeri pari sono in posizione: ');
disp(pos_pari);
```

Esercizio: Vettori e funzioni base

```
% Verificare se esiste un numero dispari
```

Esercizio: Vettori e funzioni base

```
% Verificare se esiste un numero dispari
if any(vett_mod),
    disp('esiste almeno un numero dispari');

    % Contare i numeri dispari se esistono e dire in che posizioni sono
```

```
else
    disp('non esistono numeri dispari');
end
```

Esercizio: Vettori e funzioni base

```
% Verificare se esiste un numero dispari
if any(vett_mod),
    disp('esiste almeno un numero dispari');

    % Contare i numeri dispari se esistono e dire in che posizioni sono
    count_dispari = sum(vett_mod);
    disp(['i numeri dispari sono ' num2str(count_dispari)]);


else
    disp('non esistono numeri dispari');
end
```

Esercizio: Vettori e funzioni base

```
% Verificare se esiste un numero dispari
if any(vett_mod),
    disp('esiste almeno un numero dispari');

    % Contare i numeri dispari se esistono e dire in che posizioni sono
    count_dispari = sum(vett_mod);
    disp(['i numeri dispari sono ' num2str(count_dispari)]);
    pos_dispari = find(vett_mod);
    disp('i numeri dispari sono in posizione: ');
    disp(pos_dispari);
else
    disp('non esistono numeri dispari');
end
```

Agenda

~~(20’) Es1 - Vettori e funzioni base~~

(20’) Es1.2 - Vettori, vettori everywhere!

(10’) Es2 - Funzioni e stringhe

(20’) Es3 - Files, matrici e funzioni

(20’) Es4 - Maschere di bit (ordinamento v1.0)

(20’) Es5 - Maschere di bit (ordinamento v2.0)

(20’) Es6 - Struct (rilievi altimetrici)

(20’) Es7 - Struct (film)

Esercizio: Vettori, vettori everywhere!

NON FATTO

- Scrivere un programma che letti da tastiera due vettori numerici scambi gli elementi di indice pari del primo vettore con quelli di indice dispari del secondo.

NOTA: Si continui a chiedere all'utente di inserire dei vettori fino a quando questi non soddisfano le condizioni necessarie per risolvere l'esercizio.

Esercizio: Vettori, vettori everywhere!

NON FATTO

```
v1(2:2:end) = v2(1:2:end);
```

Esercizio: Vettori, vettori everywhere!

NON FATTO

```
v1(2:2:end) = v2(1:2:end);
v2(1:2:end) =
```

Esercizio: Vettori, vettori everywhere!

NON FATTO

```
temp = v1(2:2:end);
v1(2:2:end) = v2(1:2:end);
v2(1:2:end) = temp;
```

Esercizio: Vettori, vettori everywhere!

NON FATTO

```
while
    v1 = input('Inserisci il primo vettore (tra []) : ');
    v2 = input('Inserisci il secondo vettore (tra []) : ');
```

```
end

temp = v1(2:2:end);
v1(2:2:end) = v2(1:2:end);
v2(1:2:end) = temp;

v1
v2
```

Esercizio: Vettori, vettori everywhere!

NON FATTO

```
while
    v1 = input('Inserisci il primo vettore (tra []) : ');
    v2 = input('Inserisci il secondo vettore (tra []) : ');

    pari = mod(size(v1, 2), 2) == 0 & mod(size(v2, 2), 2) == 0;
end

temp = v1(2:2:end);
v1(2:2:end) = v2(1:2:end);
v2(1:2:end) = temp;

v1
v2
```

Esercizio: Vettori, vettori everywhere!

NON FATTO

```
while
    v1 = input('Inserisci il primo vettore (tra []) : ');
    v2 = input('Inserisci il secondo vettore (tra []) : ');

    pari = mod(size(v1, 2), 2) == 0 & mod(size(v2, 2), 2) == 0;
    vettore = size(v1, 1) == 1 & size(v2, 1) == 1;

end

temp = v1(2:2:end);
v1(2:2:end) = v2(1:2:end);
v2(1:2:end) = temp;

v1
v2
```

Esercizio: Vettori, vettori everywhere!

NON FATTO

```
while
    v1 = input('Inserisci il primo vettore (tra []) : ');
    v2 = input('Inserisci il secondo vettore (tra []) : ');

    pari = mod(size(v1, 2), 2) == 0 & mod(size(v2, 2), 2) == 0;
    vettore = size(v1, 1) == 1 & size(v2, 1) == 1;
    numerico = isnumeric(v1) & isnumeric(v2);
end

temp = v1(2:2:end);
v1(2:2:end) = v2(1:2:end);
v2(1:2:end) = temp;

v1
v2
```

Esercizio: Vettori, vettori everywhere!

NON FATTO

```
while(pari == false || vettore == false || numerico == false)
    v1 = input('Inserisci il primo vettore (tra []) : ');
    v2 = input('Inserisci il secondo vettore (tra []) : ');

    pari = mod(size(v1, 2), 2) == 0 & mod(size(v2, 2), 2) == 0;
    vettore = size(v1, 1) == 1 & size(v2, 1) == 1;
    numerico = isnumeric(v1) & isnumeric(v2);
end

temp = v1(2:2:end);
v1(2:2:end) = v2(1:2:end);
v2(1:2:end) = temp;

v1
v2
```

Esercizio: Vettori, vettori everywhere!

NON FATTO

```
pari = false;
vettore = false;
numerico = false

while(pari == false || vettore == false || numerico == false)
    v1 = input('Inserisci il primo vettore (tra []) : ');
    v2 = input('Inserisci il secondo vettore (tra []) : ');

    pari = mod(size(v1, 2), 2) == 0 & mod(size(v2, 2), 2) == 0;
    vettore = size(v1, 1) == 1 & size(v2, 1) == 1;
    numerico = isnumeric(v1) & isnumeric(v2);
end

temp = v1(2:2:end);
v1(2:2:end) = v2(1:2:end);
v2(1:2:end) = temp;

v1
v2
```

Agenda

~~(20’) Es1 - Vettori e funzioni base~~

~~(20’) Es1.2 - Vettori, vettori everywhere!~~

(10’) Es2 - Funzioni e stringhe

(20’) Es3 - Files, matrici e funzioni

(20’) Es4 - Maschere di bit (ordinamento v1.0)

(20’) Es5 - Maschere di bit (ordinamento v2.0)

(20’) Es6 - Struct (rilievi altimetrici)

(20’) Es7 - Struct (film)

Esercizio: Funzioni e stringhe

Scrivere una funzione che riceve in input una stringa e restituisce **true** se è *palindroma*, **false** altrimenti.

Scrivere anche un esempio di chiamata della funzione.

Esercizio: Funzioni e stringhe

palindroma.m

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

Esercizio: Funzioni e stringhe

palindroma.m

```
1   function ris=palindroma(stringa)
2       % Ricavo la stringa invertita
3 -
4
5
6 -
7 -
8 -
9 -
10 -
```

Esercizio: Funzioni e stringhe

palindroma.m

```
1   function ris=palindroma(stringa)
2       % Ricavo la stringa invertita
3 -     stringa_r = stringa(end:-1:1);
4
5       % Se le stringhe sono uguali, la stringa palindroma
6 -
7 -
8 -
9 -
10 -
```

Esercizio: Funzioni e stringhe

palindroma.m

```
1   function ris=palindroma(stringa)
2     % Ricavo la stringa invertita
3 -   stringa_r = stringa(end:-1:1);
4
5     % Se le stringhe sono uguali, la stringa palindroma
6 -   if all(stringa == stringa_r)
7 -     ris=true;
8 -   else
9 -     ris=false;
10 -  end
```

Esercizio: Funzioni e stringhe

palindroma.m

```
1   function ris=palindroma(stringa)
2     % Ricavo la stringa invertita
3 -   stringa_r = stringa(end:-1:1);
4
5     % Se le stringhe sono uguali, la stringa palindroma
6 -   if all(stringa == stringa_r)
7 -     ris=true;
8 -   else
9 -     ris=false;
10 -  end
```

Scrivere anche un esempio di chiamata della funzione.

Esercizio: Funzioni e stringhe

palindroma.m

```
1   function ris=palindroma(stringa)
2     % Ricavo la stringa invertita
3 -   stringa_r = stringa(end:-1:1);
4
5     % Se le stringhe sono uguali, la stringa palindroma
6 -   if all(stringa == stringa_r)
7 -     ris=true;
8 -   else
9 -     ris=false;
10 -  end
```

Scrivere anche un esempio di chiamata della funzione.

Esercizio: Funzioni e stringhe

palindroma.m

```
1   function ris=palindroma(stringa)
2     % Ricavo la stringa invertita
3 -   stringa_r = stringa(end:-1:1);
4
5     % Se le stringhe sono uguali, la stringa palindroma
6 -   if all(stringa == stringa_r)
7 -     ris=true;
8 -   else
9 -     ris=false;
10 -  end
```

Scrivere anche un esempio di chiamata della funzione.

```
palindroma('ingegni')

palindroma('itopinonavevanonipoti')
```

Agenda

~~(20’) Es1 - Vettori e funzioni base~~

~~(20’) Es1.2 - Vettori, vettori everywhere!~~

~~(10’) Es2 - Funzioni e stringhe~~

(20’) Es3 - Files, matrici e funzioni

(20’) Es4 - Maschere di bit (ordinamento v1.0)

(20’) Es5 - Maschere di bit (ordinamento v2.0)

(20’) Es6 - Struct (rilievi altimetrici)

(20’) Es7 - Struct (film)

Esercizio: Files, matrici e funzioni

Esercizio: Files, matrici e funzioni

La **variabile prezzi** contiene le informazioni riguardanti I prezzi della benzina per una serie di compagnie nel mese di Settembre. Il **file** contiene una **matrice prezzi NxM** dove *N* indica il *giorno* del mese in cui e' stato registrato il prezzo, mentre *M* e' l'indice che identifica la *compagnia*. Il valore prezzi(4,3) conterra' quindi il costo della benzina per il giorno 4 presso la compagnia 3.

Esercizio: Files, matrici e funzioni

La **variabile prezzi** contiene le informazioni riguardanti I prezzi della benzina per una serie di compagnie nel mese di Settembre. Il **file** contiene una **matrice prezzi NxM** dove *N* indica il *giorno* del mese in cui e' stato registrato il prezzo, mentre *M* e' l'indice che identifica la *compagnia*. Il valore prezzi(4,3) conterra' quindi il costo della benzina per il giorno 4 presso la compagnia 3.

Esercizio: Files, matrici e funzioni

La **variabile prezzi** contiene le informazioni riguardanti I prezzi della benzina per una serie di compagnie nel mese di Settembre. Il **file** contiene una **matrice prezzi NxM** dove *N* indica il *giorno* del mese in cui e' stato registrato il prezzo, mentre *M* e' l'indice che identifica la *compagnia*. Il valore prezzi(4,3) conterra' quindi il costo della benzina per il giorno 4 presso la compagnia 3.

- Scrivere un programma Matlab che carichi la variabile prezzi contenuta in un file di nome 'prezzi.mat'.

Esercizio: Files, matrici e funzioni

La **variabile prezzi** contiene le informazioni riguardanti I prezzi della benzina per una serie di compagnie nel mese di Settembre. Il **file** contiene una **matrice prezzi NxM** dove *N* indica il *giorno* del mese in cui e' stato registrato il prezzo, mentre *M* e' l'indice che identifica la *compagnia*. Il valore prezzi(4,3) conterra' quindi il costo della benzina per il giorno 4 presso la compagnia 3.

- Scrivere un programma Matlab che carichi la variabile prezzi contenuta in un file di nome 'prezzi.mat'.
- Trovare il vettore che contenga i prezzi praticati dalle compagnie durante il primo giorno del mese:

Esercizio: Files, matrici e funzioni

La **variabile prezzi** contiene le informazioni riguardanti I prezzi della benzina per una serie di compagnie nel mese di Settembre. Il **file** contiene una **matrice prezzi NxM** dove *N* indica il *giorno* del mese in cui e' stato registrato il prezzo, mentre *M* e' l'indice che identifica la *compagnia*. Il valore prezzi(4,3) conterra' quindi il costo della benzina per il giorno 4 presso la compagnia 3.

- Scrivere un programma Matlab che carichi la variabile prezzi contenuta in un file di nome 'prezzi.mat'.
- Trovare il vettore che contenga i prezzi praticati dalle compagnie durante il primo giorno del mese:
  - Qual è stato il prezzo massimo e minimo, per ciascuna compagnia, praticato durante il mese?

Esercizio: Files, matrici e funzioni

La **variabile prezzi** contiene le informazioni riguardanti I prezzi della benzina per una serie di compagnie nel mese di Settembre. Il **file** contiene una **matrice prezzi NxM** dove *N* indica il *giorno* del mese in cui e' stato registrato il prezzo, mentre *M* e' l'indice che identifica la *compagnia*. Il valore prezzi(4,3) conterra' quindi il costo della benzina per il giorno 4 presso la compagnia 3.

- Scrivere un programma Matlab che carichi la variabile prezzi contenuta in un file di nome 'prezzi.mat'.
- Trovare il vettore che contenga i prezzi praticati dalle compagnie durante il primo giorno del mese:
  - Qual è stato il prezzo massimo e minimo, per ciascuna compagnia, praticato durante il mese?
  - Qual è stato, per ciascun giorno del mese, il prezzo massimo (e minimo) a cui trovare la benzina?

Esercizio: Files, matrici e funzioni

La **variabile prezzi** contiene le informazioni riguardanti I prezzi della benzina per una serie di compagnie nel mese di Settembre. Il **file** contiene una **matrice prezzi NxM** dove *N* indica il *giorno* del mese in cui e' stato registrato il prezzo, mentre *M* e' l'indice che identifica la *compagnia*. Il valore prezzi(4,3) conterra' quindi il costo della benzina per il giorno 4 presso la compagnia 3.

- Scrivere un programma Matlab che carichi la variabile prezzi contenuta in un file di nome 'prezzi.mat'.
- Trovare il vettore che contenga i prezzi praticati dalle compagnie durante il primo giorno del mese:
  - Qual è stato il prezzo massimo e minimo, per ciascuna compagnia, praticato durante il mese?
  - Qual è stato, per ciascun giorno del mese, il prezzo massimo (e minimo) a cui trovare la benzina?
  - Calcolare quanto è variato nel corso del mese il prezzo praticato dalle compagnie

Esercizio: Files, matrici e funzioni

La **variabile prezzi** contiene le informazioni riguardanti I prezzi della benzina per una serie di compagnie nel mese di Settembre. Il **file** contiene una **matrice prezzi NxM** dove *N* indica il *giorno* del mese in cui e' stato registrato il prezzo, mentre *M* e' l'indice che identifica la *compagnia*. Il valore prezzi(4,3) conterra' quindi il costo della benzina per il giorno 4 presso la compagnia 3.

- Scrivere un programma Matlab che carichi la variabile prezzi contenuta in un file di nome 'prezzi.mat'.
- Trovare il vettore che contenga i prezzi praticati dalle compagnie durante il primo giorno del mese:
  - Qual è stato il prezzo massimo e minimo, per ciascuna compagnia, praticato durante il mese?
  - Qual è stato, per ciascun giorno del mese, il prezzo massimo (e minimo) a cui trovare la benzina?
  - Calcolare quanto è variato nel corso del mese il prezzo praticato dalle compagnie
  - Qual è la compagnia che durante il mese ha aumentato maggiormente il prezzo, e di quanto?

Esercizio: Files, matrici e funzioni

La **variabile prezzi** contiene le informazioni riguardanti I prezzi della benzina per una serie di compagnie nel mese di Settembre. Il **file** contiene una **matrice prezzi NxM** dove *N* indica il *giorno* del mese in cui e' stato registrato il prezzo, mentre *M* e' l'indice che identifica la *compagnia*. Il valore prezzi(4,3) conterra' quindi il costo della benzina per il giorno 4 presso la compagnia 3.

- Scrivere un programma Matlab che carichi la variabile prezzi contenuta in un file di nome 'prezzi.mat'.
- Trovare il vettore che contenga i prezzi praticati dalle compagnie durante il primo giorno del mese:
  - Qual è stato il prezzo massimo e minimo, per ciascuna compagnia, praticato durante il mese?
  - Qual è stato, per ciascun giorno del mese, il prezzo massimo (e minimo) a cui trovare la benzina?
  - Calcolare quanto è variato nel corso del mese il prezzo praticato dalle compagnie
  - Qual è la compagnia che durante il mese ha aumentato maggiormente il prezzo, e di quanto?

Esercizio: Files, matrici e funzioni

La **variabile prezzi** contiene le informazioni riguardanti I prezzi della benzina per una serie di compagnie nel mese di Settembre. Il **file** contiene una **matrice prezzi NxM** dove *N* indica il *giorno* del mese in cui e' stato registrato il prezzo, mentre *M* e' l'indice che identifica la *compagnia*. Il valore prezzi(4,3) conterra' quindi il costo della benzina per il giorno 4 presso la compagnia 3.

- Scrivere un programma Matlab che carichi la variabile prezzi contenuta in un file di nome 'prezzi.mat'.
- Trovare il vettore che contenga i prezzi praticati dalle compagnie durante il primo giorno del mese:
  - Qual è stato il prezzo massimo e minimo, per ciascuna compagnia, praticato durante il mese?
  - Qual è stato, per ciascun giorno del mese, il prezzo massimo (e minimo) a cui trovare la benzina?
  - Calcolare quanto è variato nel corso del mese il prezzo praticato dalle compagnie
  - Qual è la compagnia che durante il mese ha aumentato maggiormente il prezzo, e di quanto?

- Scrivere una funzione che riceva in ingresso la variabile prezzi e la variabile g (rappresentante un giorno del mese) e restituisca la compagnia/le compagnie che erano più convenienti in quello specifico giorno e il prezzo.

Esercizio: Files, matrici e funzioni

```
% Carico il file che contiene la definizione dell'array prezzi
load 'prezzi'
```

- Trovare il vettore che contenga i prezzi praticati dalle compagnie durante il primo giorno del mese

Esercizio: Files, matrici e funzioni

```
% Carico il file che contiene la definizione dell'array prezzi
load 'prezzi'

prezzi1 = prezzi(1,:)
```

- Qual è stato il prezzo massimo e minimo, per ciascuna compagnia, praticato durante il mese?
- Qual è stato, per ciascun giorno del mese, il prezzo massimo (e minimo) a cui trovare la benzina?

Esercizio: Files, matrici e funzioni

```
% Carico il file che contiene la definizione dell'array prezzi
load 'prezzi'

prezzi1 = prezzi(1,:)

prezziBrandMin = min(prezzi)
prezziBrandMax = max(prezzi)

prezziGiornoMin = min(prezzi, [], 2)
prezziGiornoMax = max(prezzi, [], 2)

% Calcolo la variazione del prezzo come la differenza tra
% i prezzi minimi e massimi nel mese


% Massima variazione


% Compagnia con massima variazione
```

- **min(X,[],DIM)** *operates along the dimension* ***DIM***

Esercizio: Files, matrici e funzioni

```
% Carico il file che contiene la definizione dell'array prezzi
load 'prezzi'

prezzi1 = prezzi(1,:)

prezziBrandMin = min(prezzi)
prezziBrandMax = max(prezzi)

prezziGiornoMin = min(prezzi, [], 2)
prezziGiornoMax = max(prezzi, [], 2)

% Calcolo la variazione del prezzo come la differenza tra
% i prezzi minimi e massimi nel mese
diffBrand = prezziBrandMax - prezziBrandMin

% Massima variazione


% Compagnia con massima variazione
```

Esercizio: Files, matrici e funzioni

```
% Carico il file che contiene la definizione dell'array prezzi
load 'prezzi'

prezzi1 = prezzi(1,:)

prezziBrandMin = min(prezzi)
prezziBrandMax = max(prezzi)

prezziGiornoMin = min(prezzi, [], 2)
prezziGiornoMax = max(prezzi, [], 2)

% Calcolo la variazione del prezzo come la differenza tra
% i prezzi minimi e massimi nel mese
diffBrand = prezziBrandMax - prezziBrandMin

% Massima variazione
maxDiff = max(diffBrand)

% Compagnia con massima variazione
```

Esercizio: Files, matrici e funzioni

```
% Carico il file che contiene la definizione dell'array prezzi
load 'prezzi'

prezzi1 = prezzi(1,:)

prezziBrandMin = min(prezzi)
prezziBrandMax = max(prezzi)

prezziGiornoMin = min(prezzi, [], 2)
prezziGiornoMax = max(prezzi, [], 2)

% Calcolo la variazione del prezzo come la differenza tra
% i prezzi minimi e massimi nel mese
diffBrand = prezziBrandMax - prezziBrandMin

% Massima variazione
maxDiff = max(diffBrand)

% Compagnia con massima variazione
maxBrand = find(diffBrand == maxDiff)
```

Esercizio: Files, matrici e funzioni

- Scrivere una funzione che riceva in ingresso la variabile prezzi e la variabile g (rappresentante un giorno del mese) e restituisca la compagnia/le compagnie che erano più convenienti in quello specifico giorno e il prezzo.

```
function [minp comp] = conviene(prezzi, g)

  %trova il minimo prezzo del giorno g

  %trova la compagnia/compagnie con prezzo minimo
```

Esercizio: Files, matrici e funzioni

- Scrivere una funzione che riceva in ingresso la variabile prezzi e la variabile g (rappresentante un giorno del mese) e restituisca la compagnia/le compagnie che erano più convenienti in quello specifico giorno e il prezzo.

```
function [minp comp] = conviene(prezzi, g)

  %trova il minimo prezzo del giorno g
  minp = min(prezzi(g,:));

  %trova la compagnia/compagnie con prezzo minimo
```

Esercizio: Files, matrici e funzioni

- Scrivere una funzione che riceva in ingresso la variabile prezzi e la variabile g (rappresentante un giorno del mese) e restituisca la compagnia/le compagnie che erano più convenienti in quello specifico giorno e il prezzo.

```
function [minp comp] = conviene(prezzi, g)

  %trova il minimo prezzo del giorno g
  minp = min(prezzi(g,:));

  %trova la compagnia/compagnie con prezzo minimo
  comp = find(prezzi(g,:)==minp);
```

- Esempio di chiamata:

Esercizio: Files, matrici e funzioni

- Scrivere una funzione che riceva in ingresso la variabile prezzi e la variabile g (rappresentante un giorno del mese) e restituisca la compagnia/le compagnie che erano più convenienti in quello specifico giorno e il prezzo.

```
function [minp comp] = conviene(prezzi, g)

  %trova il minimo prezzo del giorno g
  minp = min(prezzi(g,:));

  %trova la compagnia/compagnie con prezzo minimo
  comp = find(prezzi(g,:)==minp);
```

- Esempio di chiamata:

```
g = input('inserire il giorno ');
[minp comp] = conviene(prezzi,g);
disp(['La più conveniente il giorno ' num2str(g) ' era/erano
          la compagnia/compagnie ' mat2str(comp) ' al prezzo di ' num2str(minp)])
```

Agenda

~~(20’) Es1 - Vettori e funzioni base~~

~~(20’) Es1.2 - Vettori, vettori everywhere!~~

~~(10’) Es2 - Funzioni e stringhe~~

~~(20’) Es3 - Files, matrici e funzioni~~

(20’) Es4 - Maschere di bit (ordinamento v1.0)

(20’) Es5 - Maschere di bit (ordinamento v2.0)

(20’) Es6 - Struct (rilievi altimetrici)

(20’) Es7 - Struct (film)

Esercizio: Maschere di bit (ordinamento v1)

- Si ordini un array di n elementi facendo uso delle istruzioni messe a disposizione da matlab.

Esercizio: Maschere di bit (ordinamento v1)

- Si ordini un array di n elementi facendo uso delle istruzioni messe a disposizione da matlab.

> *Algoritmo di ordinamento*: dato un array “disordinato” ed un array “ordinato”, trova il valore minimo nell’array “disordinato” e mettilo nella prima posizione libera dell’array “ordinato”, quindi rimuovilo dall’array “disordinato”. Itera fino a che “disordinato” é lungo 0.

Esercizio: Maschere di bit (ordinamento v1)

```
a = [10 2 -6 9 2 5]
```

- dato un array “disordinato” ed un array “ordinato”
  - trova il valore minimo nell’array “disordinato”

Esercizio: Maschere di bit (ordinamento v1)

```
a = [10 2 -6 9 2 5]

curr = min(a)
pos = find(a == curr)
```

- dato un array “disordinato” ed un array “ordinato”
- trova il valore minimo nell’array “disordinato”
- quindi rimuovilo dall’array “disordinato”

Esercizio: Maschere di bit (ordinamento v1)

```
a = [10 2 -6 9 2 5]

curr = min(a)
pos = find(a == curr)

filtro_n = a==curr
filtro = ones(1,length(a)) - filtro_n
a1 = a(logical(filtro))
```

- dato un array “disordinato” ed un array “ordinato”
- trova il valore minimo nell’array “disordinato”
- quindi rimuovilo dall’array “disordinato”
- mettilo nella prima posizione libera dell’array “ordinato”

Esercizio: Maschere di bit (ordinamento v1)

```
a = [10 2 -6 9 2 5]

curr = min(a)
pos = find(a == curr)

filtro_n = a==curr
filtro = ones(1,length(a)) - filtro_n
a1 = a(logical(filtro))

a_ord = curr
```

- dato un array “disordinato” ed un array “ordinato”
- trova il valore minimo nell’array “disordinato”
- quindi rimuovilo dall’array “disordinato”
- mettilo nella prima posizione libera dell’array “ordinato”
- Itera fino a che “disordinato” é lungo 0.

Esercizio: Maschere di bit (ordinamento v1)

```
a = [10 2 -6 9 2 5]

curr = min(a)
pos = find(a == curr)

filtro_n = a==curr
filtro = ones(1,length(a)) - filtro_n
a1 = a(logical(filtro))

a_ord = curr

while length(a1) > 0




end

a_ord
```

- dato un array “disordinato” ed un array “ordinato”
- trova il valore minimo nell’array “disordinato”
- quindi rimuovilo dall’array “disordinato”
- mettilo nella prima posizione libera dell’array “ordinato”
- Itera fino a che “disordinato” é lungo 0.

Esercizio: Maschere di bit (ordinamento v1)

```
a = [10 2 -6 9 2 5]

curr = min(a)
pos = find(a == curr)

filtro_n = a==curr
filtro = ones(1,length(a)) - filtro_n
a1 = a(logical(filtro))

a_ord = curr
```

- dato un array “disordinato” ed un array “ordinato”
- trova il valore minimo nell’array “disordinato”
- quindi rimuovilo dall’array “disordinato”
- mettilo nella prima posizione libera dell’array “ordinato”

```
while length(a1) > 0
      curr = min(a1)




end

a_ord
```

- Itera fino a che “disordinato” é lungo 0.

Esercizio: Maschere di bit (ordinamento v1)

```
a = [10 2 -6 9 2 5]

curr = min(a)
pos = find(a == curr)

filtro_n = a==curr
filtro = ones(1,length(a)) - filtro_n
a1 = a(logical(filtro))

a_ord = curr
```

- dato un array “disordinato” ed un array “ordinato”
- trova il valore minimo nell’array “disordinato”
- quindi rimuovilo dall’array “disordinato”
- mettilo nella prima posizione libera dell’array “ordinato”

```
while length(a1) > 0
      curr = min(a1)
      pos = find(a1 == curr)

      filtro_n = a1==curr
      filtro = ones(1,length(a1)) - filtro_n
      a1 = a1(logical(filtro))

      a_ord = [a_ord; curr]
end

a_ord
```

- Itera fino a che “disordinato” é lungo 0.

Esercizio: Maschere di bit (ordinamento v1)

```
a = [10 2 -6 9 2 5]

curr = min(a)
pos = find(a == curr)

filtro_n = a==curr
filtro = ones(1,length(a)) - filtro_n
a1 = a(logical(filtro))

a_ord = curr
```

- dato un array “disordinato” ed un array “ordinato”
- trova il valore minimo nell’array “disordinato”
- quindi rimuovilo dall’array “disordinato”
- mettilo nella prima posizione libera dell’array “ordinato”

```
while length(a1) > 0
      curr = min(a1)
      pos = find(a1 == curr)

      filtro_n = a1==curr
      filtro = ones(1,length(a1)) - filtro_n
      a1 = a1(logical(filtro))

      a_ord = [a_ord; curr]
end

a_ord
```

- Itera fino a che “disordinato” é lungo 0.

Agenda

~~(20’) Es1 - Vettori e funzioni base~~

~~(20’) Es1.2 - Vettori, vettori everywhere!~~

~~(10’) Es2 - Funzioni e stringhe~~

~~(20’) Es3 - Files, matrici e funzioni~~

~~(20’) Es4 - Maschere di bit (ordinamento v1.0)~~

(20’) Es5 - Maschere di bit (ordinamento v2.0)

(20’) Es6 - Struct (rilievi altimetrici)

(20’) Es7 - Struct (film)

Esercizio: Maschere di bit (ordinamento

NON FATTO

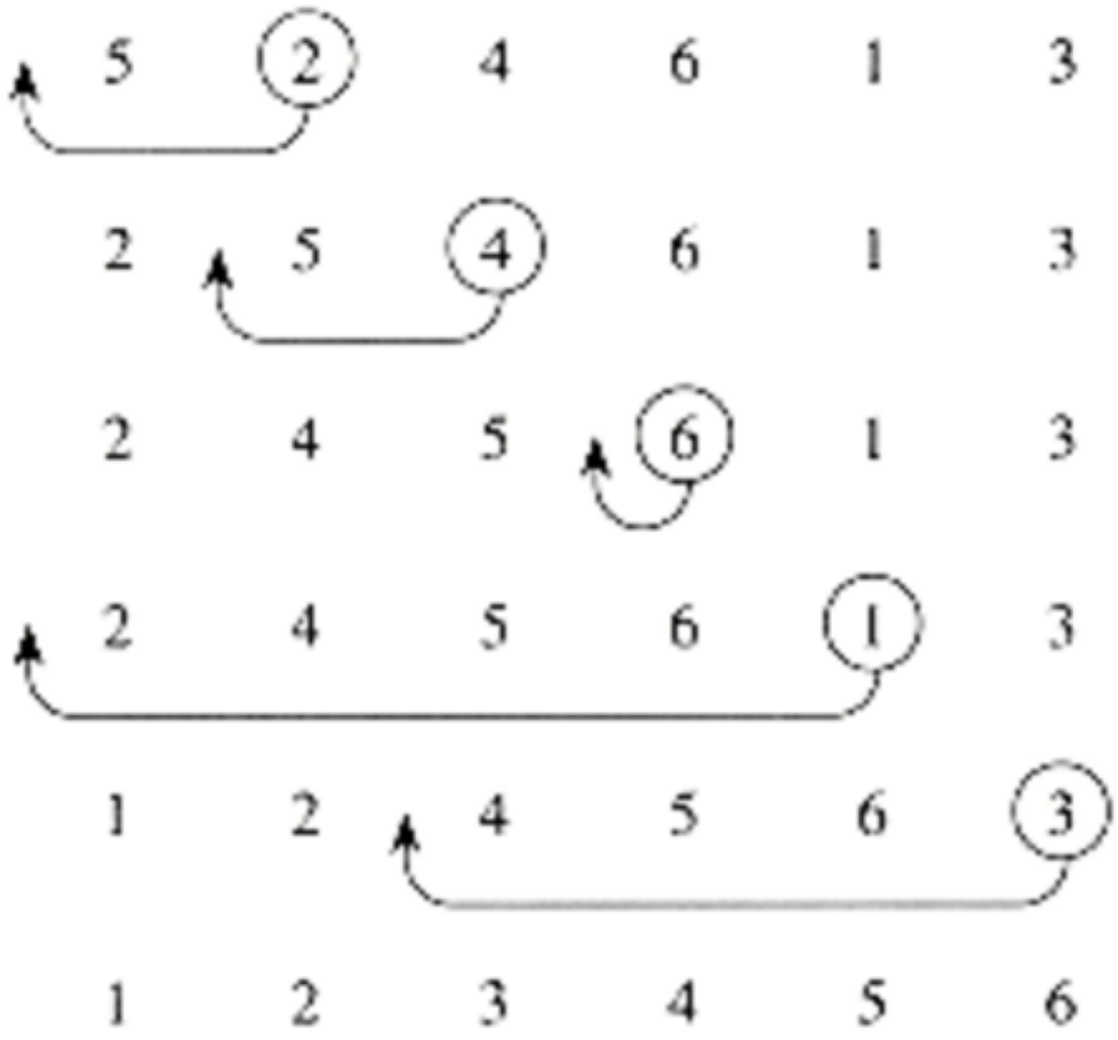
- Si ordini un array di n elementi facendo uso delle istruzioni messe a disposizione da matlab.

Esercizio: Maschere di bit (ordinamento

NON FATTO

- Si ordini un array di n elementi facendo uso delle istruzioni messe a disposizione da matlab.

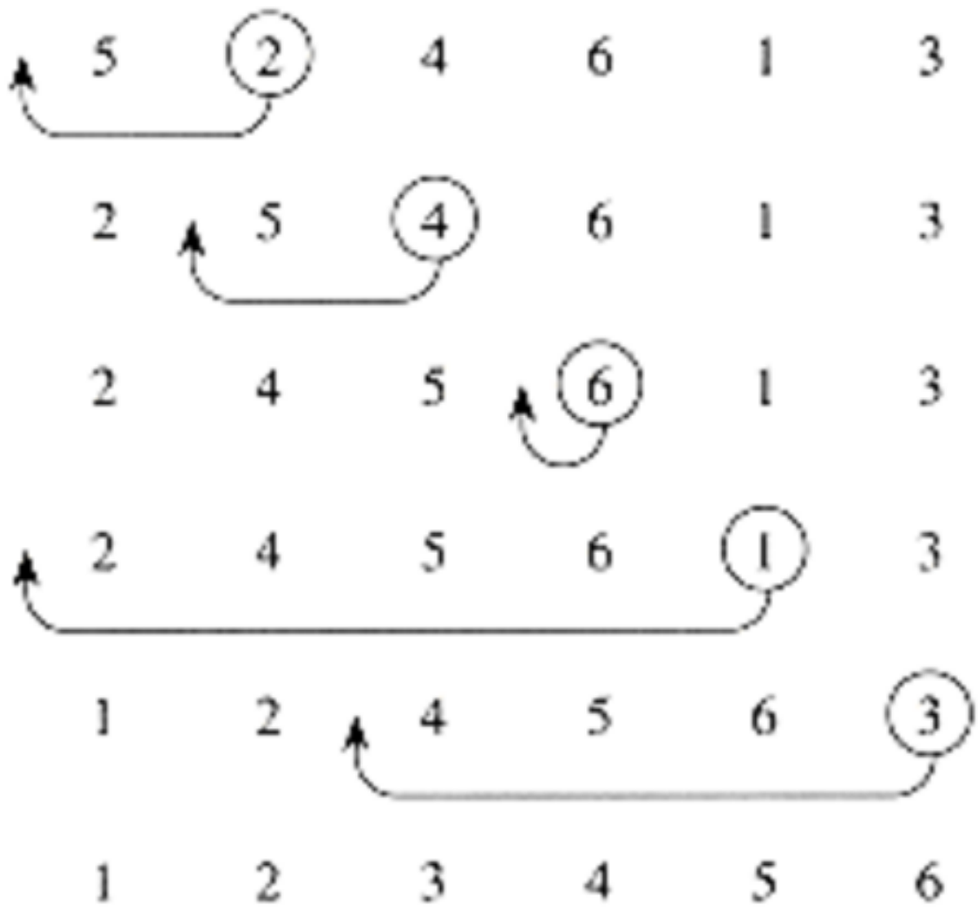
Algoritmo di ordinamento: “Insertion sort”
(https://it.wikipedia.org/wiki/Insertion_sort)



Esercizio: Maschere di bit (ordinamento

NON FATTO

```
a = [5 2 4 6 1 3]
```



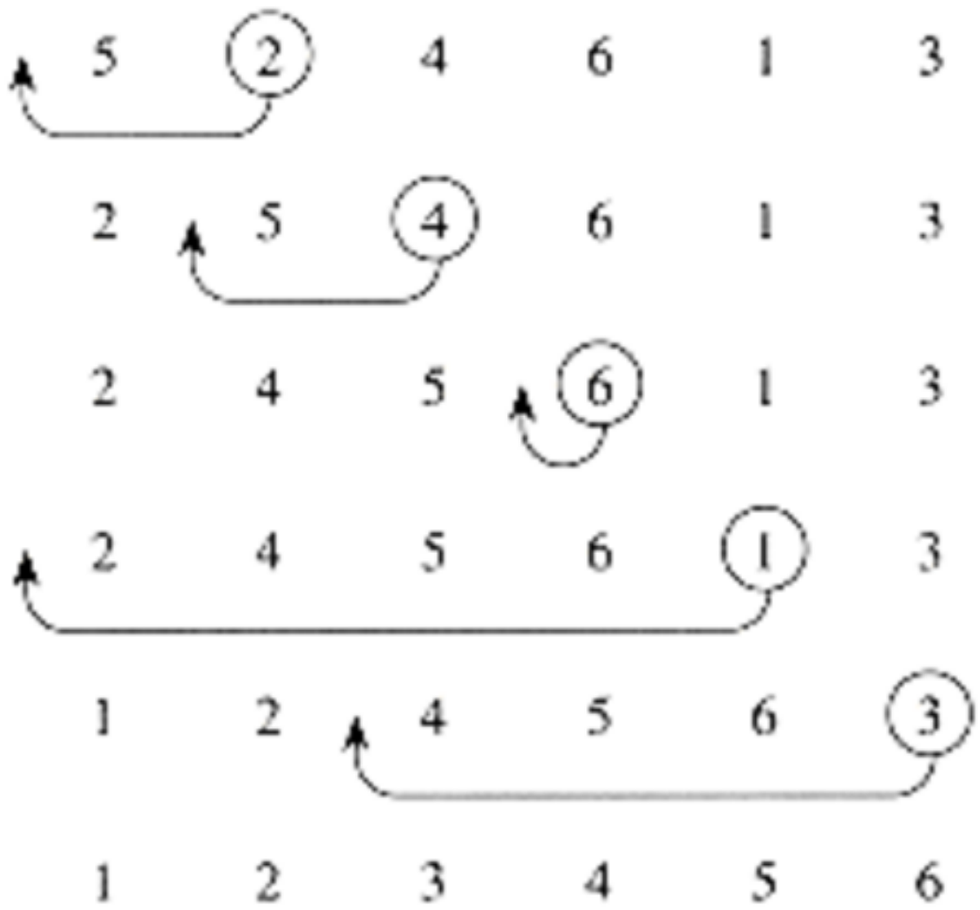
Esercizio: Maschere di bit (ordinamento

NON FATTO

```
a = [5 2 4 6 1 3]

a_ord = a(1);
for i=2:length(a)
    a_ord = inserisci(a_ord,a(i));
end

a_ord
```



Esercizio: Maschere di bit (ordinamento

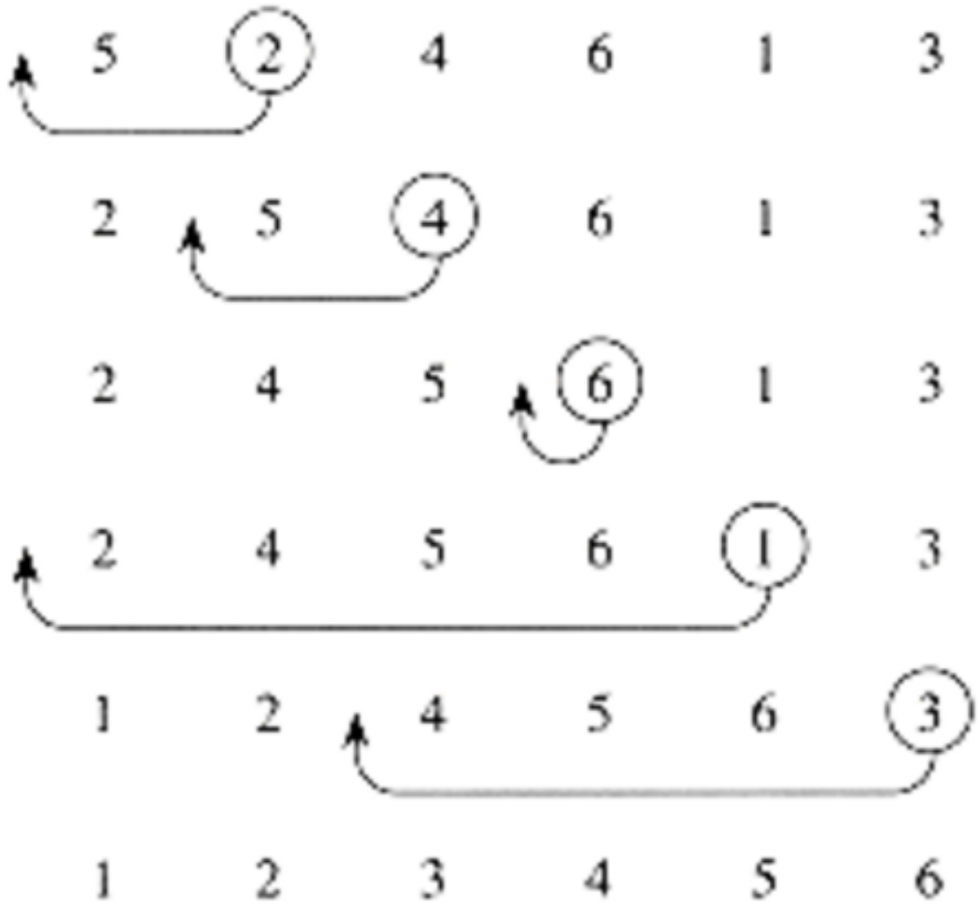
NON FATTO

```
a = [5 2 4 6 1 3]

a_ord = a(1);
for i=2:length(a)
    a_ord = inserisci(a_ord,a(i));
end

a_ord
```

```
function v_ord = inserisci(v,e)
    v_ord = [v(v<=e) e v(v>e)];
```



Da tenere sempre a mente…

NON FATTO

Da tenere sempre a mente…

NON FATTO

In MATLAB si ragiona con un paradigma diverso dal C:

Da tenere sempre a mente…

NON FATTO

In MATLAB si ragiona con un paradigma diverso dal C:

- tutto è un array;

Da tenere sempre a mente…

NON FATTO

In MATLAB si ragiona con un paradigma diverso dal C:

- tutto è un array;
- gli **scalari** sono un caso particolare di array multidimensionale;

Da tenere sempre a mente…

NON FATTO

In MATLAB si ragiona con un paradigma diverso dal C:

- tutto è un array;
- gli **scalari** sono un caso particolare di array multidimensionale;
- le funzioni operano **in parallelo sui dati**, contenuti in forma array multidimensionale nelle variabili;

Da tenere sempre a mente…

NON FATTO

In MATLAB si ragiona con un paradigma diverso dal C:

- tutto è un array;
- gli **scalari** sono un caso particolare di array multidimensionale;
- le funzioni operano **in parallelo sui dati**, contenuti in forma array multidimensionale nelle variabili;
- le operazioni matriciali sono esprimibili con **poche istruzioni** (al limite una sola istruzione)

Da tenere sempre a mente…

NON FATTO

In MATLAB si ragiona con un paradigma diverso dal C:

- tutto è un array;
- gli **scalari** sono un caso particolare di array multidimensionale;
- le funzioni operano **in parallelo sui dati**, contenuti in forma array multidimensionale nelle variabili;
- le operazioni matriciali sono esprimibili con **poche istruzioni** (al limite una sola istruzione)
- i **for** sono quasi sempre da **evitare**

Da tenere sempre a mente…

NON FATTO

In MATLAB si ragiona con un paradigma diverso dal C:

- tutto è un array;
- gli **scalari** sono un caso particolare di array multidimensionale;
- le funzioni operano **in parallelo sui dati**, contenuti in forma array multidimensionale nelle variabili;
- le operazioni matriciali sono esprimibili con **poche istruzioni** (al limite una sola istruzione)
- i **for** sono quasi sempre da **evitare**
- le maschere di bit tornano molto comode quando dovete **filtrare** in qualche modo i dati;

Agenda

NON FATTO

~~(20’) Es1 - Vettori e funzioni base~~

~~(20’) Es1.2 - Vettori, vettori everywhere!~~

~~(10’) Es2 - Funzioni e stringhe~~

~~(20’) Es3 - Files, matrici e funzioni~~

~~(20’) Es4 - Maschere di bit (ordinamento v1.0)~~

~~(20’) Es5 - Maschere di bit (ordinamento v2.0)~~

(20’) Es6 - Struct (rilievi altimetrici)

(20’) Es7 - Struct (film)

Esercizio: Struct (rilievi altimetrici)

NON FATTO

- Si sviluppi un programma in matlab che acquisisce da tastiera i dati relativi a rilievi altimetrici e stampa a video l'altitudine media di tutti quelli che hanno latitudine compresa tra 10 e 80 e longitudine tra 30 e 60

Esercizio: Struct (rilievi altimetrici)

NON FATTO

```
more = input('vuoi inserire valori altimetrici? (s/n)');
```

Esercizio: Struct (rilievi altimetrici)

NON FATTO

```
more = input('vuoi inserire valori altimetrici? (s/n)');
ii=1;
while more=='s'



    ii = ii+1;
    more = input('vuoi inserire altri valori altimetrici? (s/n)');
end
```

Esercizio: Struct (rilievi altimetrici)

NON FATTO

```
more = input('vuoi inserire valori altimetrici? (s/n)');
ii=1;
while more=='s'
    arch(ii).altitudine = input('altitudine ');
    arch(ii).longitudine = input('longitudine ');
    arch(ii).latitudine = input('latitudine ');
    ii = ii+1;
    more = input('vuoi inserire altri valori altimetrici? (s/n)');
end
```

Esercizio: Struct (rilievi altimetrici)

NON FATTO

```
more = input('vuoi inserire valori altimetrici? (s/n)');
ii=1;
while more=='s'
    arch(ii).altitudine = input('altitudine ');
    arch(ii).longitudine = input('longitudine ');
    arch(ii).latitudine = input('latitudine ');
    ii = ii+1;
    more = input('vuoi inserire altri valori altimetrici? (s/n)');
end
```

```
jj=1;
for ii=1:length(arch)




end
disp(['la media degli elementi selezionati e` ' num2str(mean(elemSelez))])
```

Esercizio: Struct (rilievi altimetrici)

NON FATTO

```
more = input('vuoi inserire valori altimetrici? (s/n)');
ii=1;
while more=='s'
    arch(ii).altitudine = input('altitudine ');
    arch(ii).longitudine = input('longitudine ');
    arch(ii).latitudine = input('latitudine ');
    ii = ii+1;
    more = input('vuoi inserire altri valori altimetrici? (s/n)');
end
```

```
jj=1;
for ii=1:length(arch)
    % attenzione: la condizione deve essere scritta sulla stessa linea…
    if arch(ii).latitudine>=10&&arch(ii).latitudine<=80 &&
       arch(ii).longitudine>=30&&arch(ii).longitudine<=60
        elemSelez(jj) = arch(ii).altitudine;
        jj=jj+1;
    end
end
disp(['la media degli elementi selezionati e` ' num2str(mean(elemSelez))])
```

Agenda

NON FATTO

~~(20’) Es1 - Vettori e funzioni base~~

~~(20’) Es1.2 - Vettori, vettori everywhere!~~

~~(10’) Es2 - Funzioni e stringhe~~

~~(20’) Es3 - Files, matrici e funzioni~~

~~(20’) Es4 - Maschere di bit (ordinamento v1.0)~~

~~(20’) Es5 - Maschere di bit (ordinamento v2.0)~~

~~(20’) Es6 - Struct (rilievi altimetrici)~~

(20’) Es7 - Struct (film)

Esercizio: Struct (film)

NON FATTO

- Si vuole compilare la pagella dei propri film preferiti. Per farlo:
  - Scrivere un programma che chieda di inserire i film. Ogni film e' caratterizzato da un anno, un titolo e un voto;
  - Scrivere il codice che permetta di visualizzare il numero totale di film con voto superiore a 6;
  - Scrivere il codice che permetta di visualizzare i titoli ed i voti dei film prodotti tra il 2000 e il 2005;

Esercizio: Struct (film)

NON FATTO

```
% Inserimento films
```

Esercizio: Struct (film)

NON FATTO

```
% Inserimento films

stopInput='S';
count=1;

while(stopInput=='S')
```

```
end
```

Esercizio: Struct (film)

NON FATTO

```
% Inserimento films

stopInput='S';
count=1;

while(stopInput=='S')

    films(count).titolo = input('Inserisci il titolo: ');
    films(count).anno = input('Inserisci anno: ');
    films(count).voto = input('Inserisci il voto: ');




end
```

Esercizio: Struct (film)

NON FATTO

```
% Inserimento films

stopInput='S';
count=1;

while(stopInput=='S')

    films(count).titolo = input('Inserisci il titolo: ');
    films(count).anno = input('Inserisci anno: ');
    films(count).voto = input('Inserisci il voto: ');

    count = count + 1;

    stopInput = input('Vuoi inserire altri film? (S/N) ');

end
```

Esercizio: Struct (film)

NON FATTO

```
% Numero totale di film con voto superiore a 6
```

Esercizio: Struct (film)

NON FATTO

```
% Numero totale di film con voto superiore a 6
sum([films.voto] > 6)

% Film prodotti tra il 2002 ed il 2005
```

Esercizio: Struct (film)

NON FATTO

```
% Numero totale di film con voto superiore a 6
sum([films.voto] > 6)

% Film prodotti tra il 2002 ed il 2005
idx = find([films.anno] > 2002 & [films.anno] < 2005);

% Stampa titoli e voti separatamente
```

Esercizio: Struct (film)

NON FATTO

```
% Numero totale di film con voto superiore a 6
sum([films.voto] > 6)

% Film prodotti tra il 2002 ed il 2005
idx = find([films.anno] > 2002 & [films.anno] < 2005);

% Stampa titoli e voti separatamente
films(idx).titolo
films(idx).voto

% Stampa titolo e voto affiancati
```

Esercizio: Struct (film)

NON FATTO

```
% Numero totale di film con voto superiore a 6
sum([films.voto] > 6)

% Film prodotti tra il 2002 ed il 2005
idx = find([films.anno] > 2002 & [films.anno] < 2005);

% Stampa titoli e voti separatamente
films(idx).titolo
films(idx).voto

% Stampa titolo e voto affiancati
for i=idx

end
```

Esercizio: Struct (film)

NON FATTO

```
% Numero totale di film con voto superiore a 6
sum([films.voto] > 6)

% Film prodotti tra il 2002 ed il 2005
idx = find([films.anno] > 2002 & [films.anno] < 2005);

% Stampa titoli e voti separatamente
films(idx).titolo
films(idx).voto

% Stampa titolo e voto affiancati
for i=idx
    display([films(i).titolo ' ' num2str(films(i).voto)])
end
```

Agenda

NON FATTO

~~(20') Es1 - Vettori e funzioni base~~

~~(20') Es1.2 - Vettori, vettori everywhere!~~

~~(10') Es2 - Funzioni e stringhe~~

~~(20') Es3 - Files, matrici e funzioni~~

~~(20') Es4 - Maschere di bit (ordinamento v1.0)~~

~~(20') Es5 - Maschere di bit (ordinamento v2.0)~~

~~(20') Es6 - Struct (rilievi altimetrici)~~

~~(20') Es7 - Struct (film)~~