

 POLITECNICO DI MILANO

Dipartimento di  
Elettronica e Informazione

# MATLAB, from zero to ~~hero~~ structs

Matteo Ferroni  
matteo.ferroni@polimi.it

11/12/2018



POLITECNICO  
DI MILANO



---

# MATLAB, from zero to ~~hero~~ structs

## 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!*

# Da tenere sempre a mente...

---

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

---

~~(10') Es1 - Vettori e funzioni base~~

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

(15') Es2 - Stringhe palindrome

(30') Es3 - Files e matrici

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

(20') Es7 - Struct (film)

# Esercizio: Stringhe palindrome

---

Scrivere uno script riceva in input una stringa e dica se è *palindroma*.

# Esercizio: Stringhe palindrome

---

```
stringa = input('Inserire una stringa: ');
```

# Esercizio: Stringhe palindrome

---

```
stringa = input('Inserire una stringa: ');  
  
% Ricavo la stringa invertita  
stringa_r = stringa(end:-1:1);
```

# Esercizio: Stringhe palindrome

---

```
stringa = input('Inserire una stringa: ');

% Ricavo la stringa invertita
stringa_r = stringa(end:-1:1);

% Se le stringhe sono uguali, la stringa palindroma
if all(stringa == stringa_r)
    disp('La stringa è palindroma!');
else
    disp('La stringa NON è palindroma!')
end
```

# Agenda

---

~~(10') Es1 - Vettori e funzioni base~~

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

~~(15') Es2 - Stringhe palindrome~~

(30') Es3 - Files e matrici

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

(20') Es7 - Struct (film)

# Esercizio: Files e matrici

---

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 è stato registrato il prezzo, mentre  $M$  è 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?
- Dato un giorno del mese, si ricavi la compagnia/le compagnie che erano più convenienti in quello specifico giorno, e il prezzo.

# Esercizio: Files e matrici

---

```
% 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 e matrici

---

```
% 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 e matrici

---

```
% 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)
```

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

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

```
% Massima variazione
```

```
% Compagnia con massima variazione
```

# Esercizio: Files e matrici

---

```
% 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 e matrici

---

```
% 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 e matrici

---

```
% 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 e matrici

---

- Dato un giorno del mese, si ricavi la compagnia/le compagnie che erano più convenienti in quello specifico giorno, e il prezzo.

```
g = input('inserire il giorno ');
```

# Esercizio: Files e matrici

---

- Dato un giorno del mese, si ricavi la compagnia/le compagnie che erano più convenienti in quello specifico giorno, e il prezzo.

```
g = input('inserire il giorno ');
```

```
%trova il minimo prezzo del giorno g
```

```
minp = min(prezzi(g,:));
```

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

```
comp = find(prezzi(g,)==minp);
```

# Esercizio: Files e matrici

---

- Dato un giorno del mese, si ricavi la compagnia/le compagnie che erano più convenienti in quello specifico giorno, e il prezzo.

```
g = input('inserire il giorno ');

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

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

disp(['La più conveniente il giorno ' num2str(g)
      ' era/erano la compagnia/compagnie ' mat2str(comp)
      ' al prezzo di ' num2str(minp)])
```

# Agenda

---

~~(10') Es1 - Vettori e funzioni base~~

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

~~(15') Es2 - Stringhe palindrome~~

~~(30') Es3 - Files e matrici~~

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

(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.

*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
```

```
while length(a1) > 0
```

```
    curr ≡ min(a1)
```

```
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
```

```
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
```

- 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.

**...manca qualcosa?**

# 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
```

```
    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*ones(length(pos),1)]
```

```
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.

***Permette di non perdere molteplici “minimi”!***

# Agenda

---

~~(10') Es1 - Vettori e funzioni base~~

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

~~(15') Es2 - Stringhe palindrome~~

~~(30') Es3 - Files e matrici~~

~~(30') Es4 - Maschere di bit (ordinamento v1.0)~~

(20') Es7 - Struct (film)

# Matrici nel mondo reale (e in C)

```
int matrice[4][4];
```

```
matrice[0][2] = 39;
```

```
int i = 2;
```

```
int j = 3;
```

```
matrice[i][j] = 42;
```

|   | 0 | 1 | 2         | 3         |
|---|---|---|-----------|-----------|
| 0 |   |   | <b>39</b> |           |
| 1 |   |   |           |           |
| 2 |   |   |           | <b>42</b> |
| 3 |   |   |           |           |

# Struct nel mondo reale (e in C)

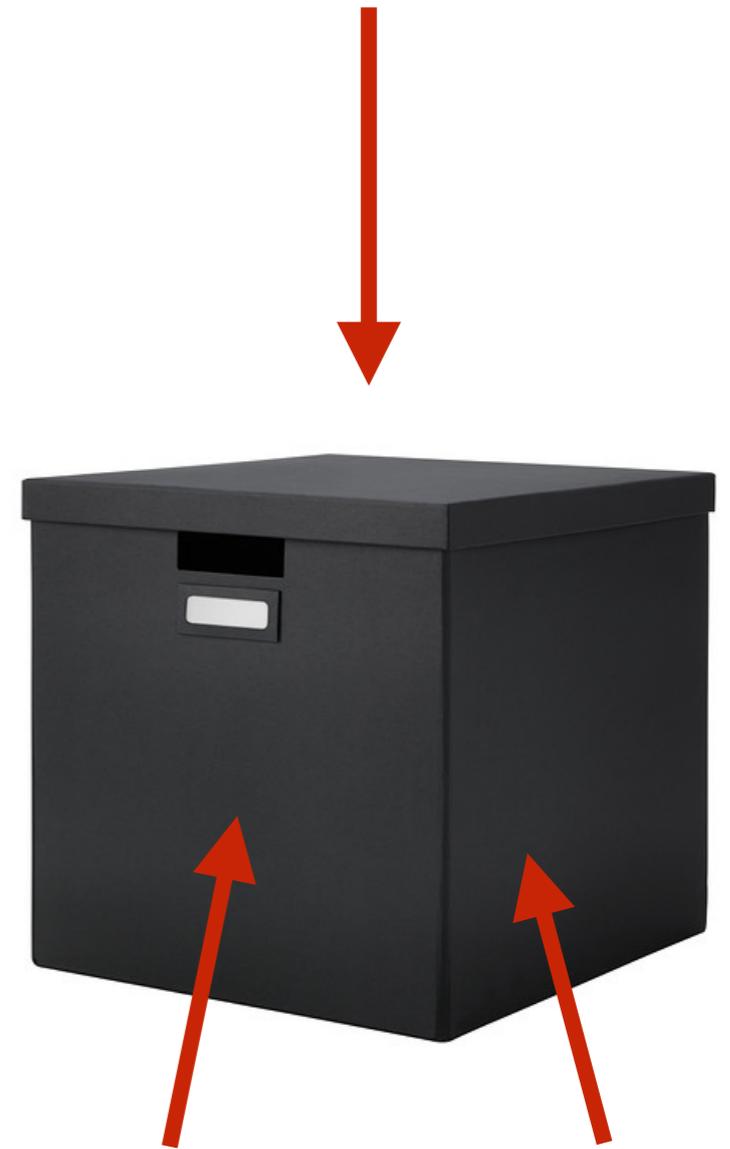
```
typedef struct {  
    int numero;  
    char stringa[10];  
} scatola;
```

```
scatola laMiaNuovaScatola;
```

```
laMiaNuovaScatola.numero = 42;
```

```
strcpy(laMiaNuovaScatola.stringa, "ciao");
```

```
int numero;  
char stringa[10];
```



**42**

**“ciao”**

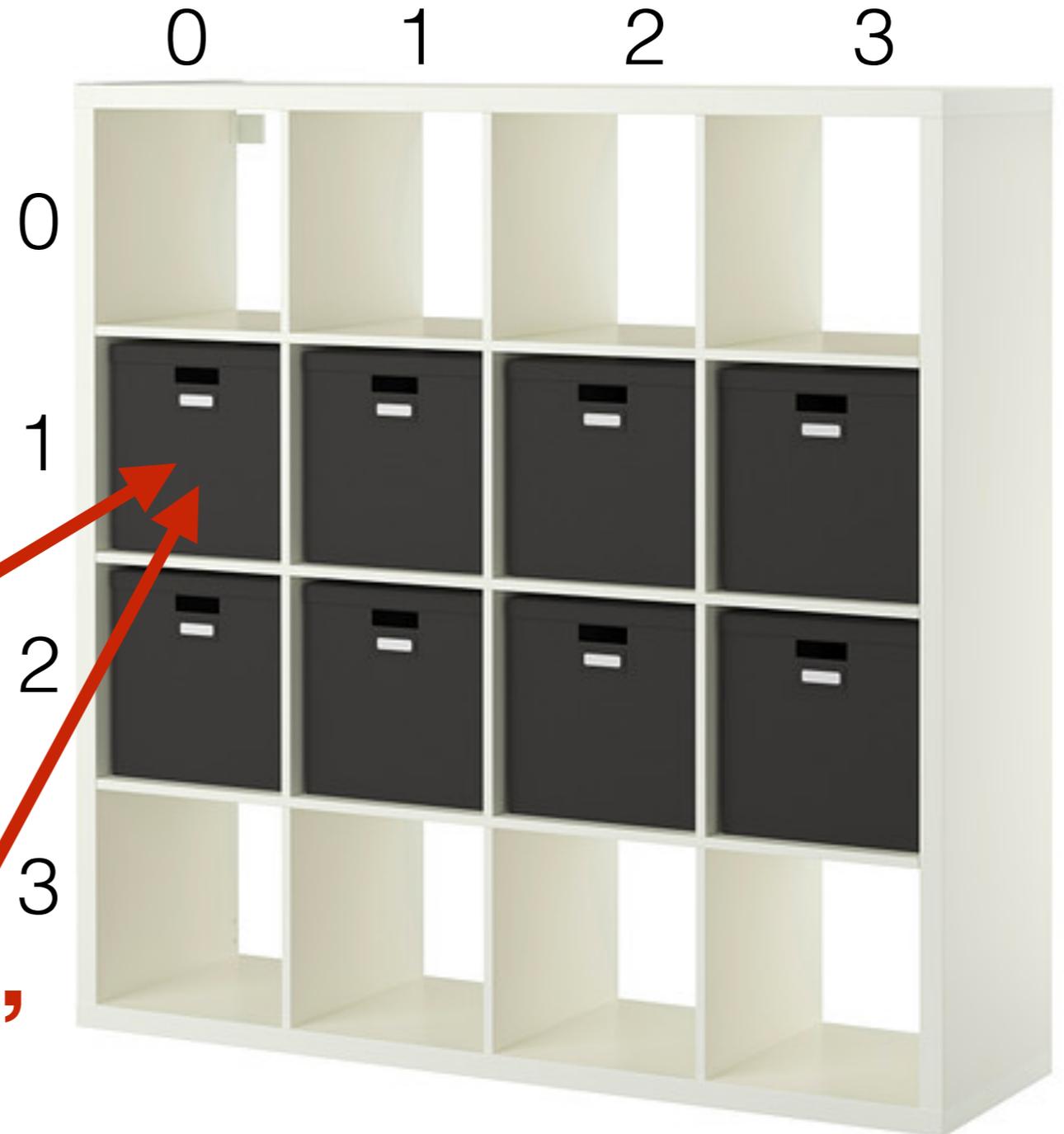
# Matrici e struct nel mondo reale (e in C)

```
typedef struct {  
    int numero;  
    char stringa[10];  
} scatola;
```

```
scatola matrice[4][4];
```

42

“ciao”



```
matrice[1][0].numero = 42;
```

```
strcpy(matrice[1][0].stringa, “ciao”);
```

# Strutture (e array di strutture)

Una **struttura** è un tipo di dato composto da elementi eterogenei

- Ogni elemento individuale è chiamato **campo** e ha un nome
- Come con gli scalari, si può passare da un elemento singolo (matrice  $1 \times 1$ ) a un vettore (matrice  $1 \times n$ )
- Ci sono due modi per creare una struttura:
  - Campo per campo mediante assegnamento
  - Tutto in una volta mediante la funzione struct



- Esempio: la struttura studente
  - `studente.nome = 'Giovanni Rossi';`
  - `studente.indirizzo = 'Via Roma 23';`
  - `studente.citta = 'Cosenza';`
  - `studente.media = 25;`
  - `whos studente`

| Name     | Size | Bytes | Class  | Attributes |
|----------|------|-------|--------|------------|
| studente | 1x1  | 568   | struct |            |

- `%aggiungo un nuovo studente... -> array 1x2`
- `studente(2).nome = 'Giulia Gatti';`
- `studente(2).media = 30;`
- Nota: quando un elemento viene definito, tutti i suoi campi sono creati e inizializzati a valore nullo (vettore vuoto [])



- Consente di preallocare una struttura o un array di strutture
  - `str_array = struct('campo1', val1, 'campo2', val2, ...)`
- Esempio

```
>> rilievoAltimetrico=struct('latitudine',20,'longitudine',30, 'altitudine', 1300)
```

```
rilievoAltimetrico =  
  latitudine: 20  
  longitudine: 30  
  altitudine: 1300
```

# Creazione di array di strutture

- Se si allunga un array assegnando un valore a una componente di indice  $>$  dimensione corrente
  - ▶ i nuovi elementi, in posizione precedente a quello inserito esplicitamente, vengono inizializzati al solito valore 'nullo' []

# Creazione di array di strutture

- Se si allunga un array assegnando un valore a una componente di indice  $>$  dimensione corrente
  - ▶ i nuovi elementi, in posizione precedente a quello inserito esplicitamente, vengono inizializzati al solito valore 'nullo' []
- Esempio
  - ▶ `rilieviAltimetrici(1000)=struct('latitudine',80,'longitudine',[], 'altitudine', 1450)`
    - `rilieviAltimetrici =`
    - 1x1000 struct array with fields
    - `latitudine`
    - `longitudine`
    - `altitudine`



# Creazione di array di strutture



RIPASSO

DIPARTIMENTO DI ELETTRONICA E INFORMAZIONE

- Se si allunga un array assegnando un valore a una componente di indice  $>$  dimensione corrente
  - ▶ i nuovi elementi, in posizione precedente a quello inserito esplicitamente, vengono inizializzati al solito valore 'nullo' []

- Esempio

- ▶ `rilieviAltimetrici(1000)=struct('latitudine',80,'longitudine',[],`  
`'altitudine', 1450)`

- `rilieviAltimetrici =`
- 1x1000 struct array with fields:
  - `latitudine`
  - `longitudine`
  - `altitudine`

Array vuoto. Attenzione: se si inserisce un valore (es. 20), questo viene assunto dal campo `longitudine` dell'elemento 1000, ma non dallo stesso campo degli altri elementi dell'array



- Aggiunta di un campo: facciamo riferimento alla definizione di studente delle slide precedenti
  - `studente(2).esami = [20 25 30];`



- Aggiunta di un campo: facciamo riferimento alla definizione di studente delle slide precedenti
  - `studente(2).esami = [20 25 30];`
- Il campo esami viene aggiunto a tutte le strutture che fanno parte di studente
  - Avrà un valore iniziale per `studente(2)`. Sarà vuoto per tutti gli altri elementi dell'array

# ↳ Uso dei dati nelle strutture

- Notazione con il “punto”, uguale al C
- Esempi
  - `studente(2).nome`
  - `studente(2).esami(2)`
  - `unNome = studente(1).nome`
  - `studente(2).indirizzo=studente(1).indirizzo`
  - `%mean` calcola la media degli elementi di un array
  - `mean(studente(2).esami)`

# ↳ Uso dei dati nelle strutture

- Notazione con il “punto”, uguale al C
- Esempi
  - `studente(2).nome`
  - `studente(2).esami(2)`
  - `unNome = studente(1).nome`
  - `studente(2).indirizzo=studente(1).indirizzo`
  - `%mean` calcola la media degli elementi di un array
  - `mean(studente(2).esami)`
- Estrazione dei valori che un campo assume in tutti gli elementi di un array di strutture (NB: ipotizziamo che le strutture dell'array *studente* abbiano un campo 'media' e che l'array abbia due componenti)
  - `a = [studente.media]`  $\longrightarrow$  `a = [25 30]`

# Array di strutture innestati

- Un campo di un array di strutture può essere di qualsiasi tipo (come in C)
- E` quindi possibile avere un campo che è, di nuovo, una struttura. Esempio
  - `studente(1).corso(1).nome= 'InformaticaB' ;`
  - `studente(1).corso(1).docente= 'Von Neumann' ;`
  - `studente(1).corso(2).nome= 'Matematica' ;`
  - `studente(1).corso(2).docente= 'Eulero' ;`



- 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

# Soluzione (1)



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



```
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 ');  
  
end
```

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

```
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
```

# Soluzione (2)



# Soluzione (2)

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

```
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  
  
        end  
    end  
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
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))]);
```

# Esercizio: Struct (film)

---

- 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)

---

```
% Inserimento films
```

# Esercizio: Struct (film)

---

```
% Inserimento films
```

```
stopInput='S';
```

```
count=1;
```

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

```
end
```

# Esercizio: Struct (film)

---

```
% 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)

---

```
% 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)

---

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

# Esercizio: Struct (film)

---

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

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

# Esercizio: Struct (film)

---

```
% 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)

---

```
% 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)

---

```
% 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)

---

```
% 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

---

~~(10') Es1 - Vettori e funzioni base~~

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

~~(15') Es2 - Stringhe palindrome~~

~~(30') Es3 - Files e matrici~~

~~(30') Es4 - Maschere di bit (ordinamento v1.0)~~

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