

 POLITECNICO DI MILANO

Dipartimento di  
Elettronica e Informazione

# MATLAB: structs and functions

Matteo Ferroni  
[matteo.ferroni@polimi.it](mailto:matteo.ferroni@polimi.it)

28/11/2017



POLITECNICO  
DI MILANO



# Agenda

---

(20') Vettori (ripasso)

(20') Funzioni e stringhe

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

(10') Pausa

(20') Maschere di bit (ordinamento v2.0)

(20') Funzioni e divisori

(30') Struct (film)

# Esercizio: Vettori e maschere

---

```
% chiedere all'utente di inserire un vettore e un numero
%
% calcolare:
% # il numero di elementi del vettore uguali al numero inserito
% # il numero di elementi del vettore maggiori del numero inserito
% # il numero di elementi del vettore minori del numero inserito
%
% indicare poi il valore di tali elementi,
% la loro posizione del vettore
% il vettore binario per ogni operazione richiesta
```

# Esercizio: Vettori e maschere

---

```
%x = input('inserire un numero')  
%vettore = input('inserisci un vettore')
```

# Esercizio: Vettori e maschere

---

```
%x = input('inserire un numero')  
%vettore = input('inserisci un vettore')
```

```
x = 10  
vettore = [10 20 30 50 60]
```

```
%vettore binario  
Buguali=  
Bmaggiori  
Bminori=
```

# Esercizio: Vettori e maschere

---

```
%x = input('inserire un numero')  
%vettore = input('inserisci un vettore')
```

```
x = 10  
vettore = [10 20 30 50 60]
```

```
%vettore binario  
Buguali= vettore==x  
Bmaggiori= vettore>x  
Bminori= vettore<x
```

```
%valore degli elementi  
Vuguali=  
Vmaggiori  
Vminori=
```

# Esercizio: Vettori e maschere

---

```
%x = input('inserire un numero')  
%vettore = input('inserisci un vettore')
```

```
x = 10  
vettore = [10 20 30 50 60]
```

```
%vettore binario
```

```
Buguali= vettore==x
```

```
Bmaggiori= vettore>x
```

```
Bminori= vettore<x
```

```
%valore degli elementi
```

```
Vuguali= vettore(vettore==x)
```

```
Vmaggiori= vettore(vettore>x)
```

```
Vminori= vettore(vettore<x)
```

# Esercizio: Vettori e maschere

---

`%numero di elementi`



# Esercizio: Vettori e maschere

---

```
%numero di elementi  
Nuguali=size( vettore (vettore==x),2)  
Nmaggiori=size( vettore (vettore>x),2)  
Nminori=size( vettore (vettore<x),2)
```

```
%alternativa:  
% Nmaggiori = sum(vettore > x)
```

```
%posizione degli elementi  
Puguali=  
Pmaggiori  
Pminori=
```

# Esercizio: Vettori e maschere

---

```
%numero di elementi  
Nuguali=size( vettore (vettore==x),2)  
Nmaggiori=size( vettore (vettore>x),2)  
Nminori=size( vettore (vettore<x),2)
```

```
%alternativa:  
% Nmaggiori = sum(vettore > x)
```

```
%posizione degli elementi  
Puguali= find(vettore==x)  
Pmaggiori= find(vettore>x)  
Pminori= find(vettore<x)
```

# Agenda

---

~~(20') Vettori (ripasso)~~

(20') Funzioni e stringhe

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

(10') Pausa

(20') Maschere di bit (ordinamento v2.0)

(20') Funzioni e divisori

(30') Struct (film)

# Funzioni

## FUNZIONI

- Hanno un proprio ambiente di esecuzione
  - variabili locali distinte da quelle del chiamante
  - variabili locali cessano di esistere al ritorno
- Comunicazione mediante copiatura parametri
- Adatte a sviluppo sistematico di applicazioni complesse
  - Unità di programma con alta coesione interna e interfacce minimali e chiaramente identificate

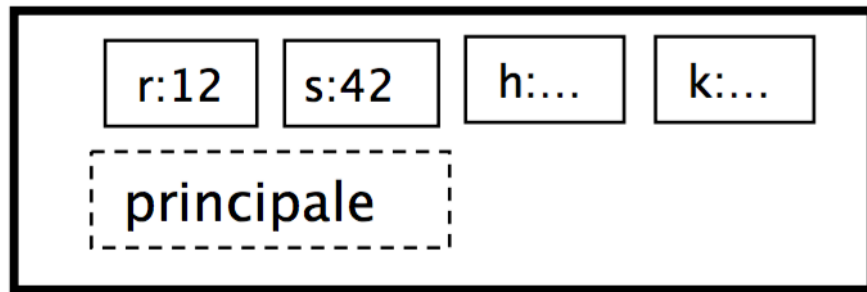
## SCRIPT

- NON hanno un proprio ambiente di esecuzione
  - variabili dello script sono le stesse del chiamante
  - variabili create nello script continuano a esistere
- Comunicazione mediante scrittura/lettura variabili comuni
- Adatti a sviluppo esplorativo e prototipale

# Funzioni

---

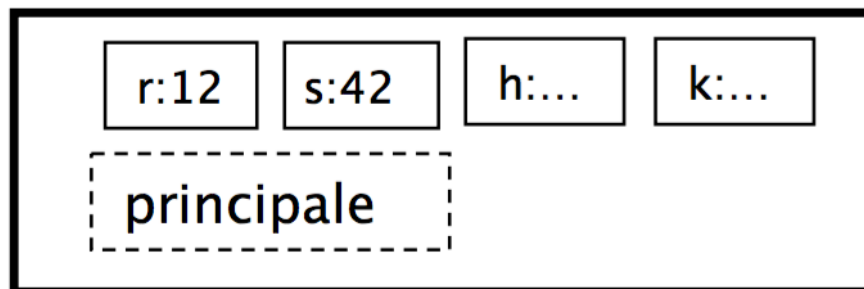
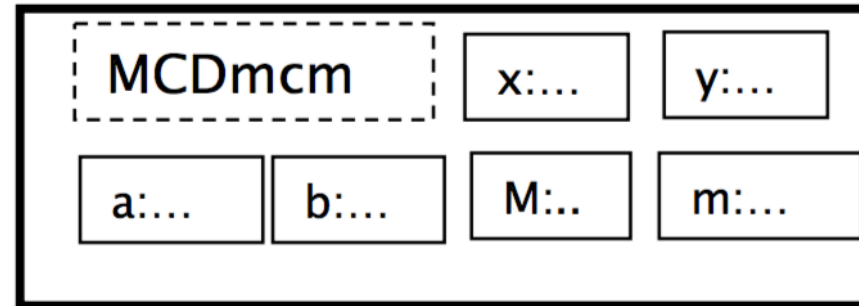
**[h,k] = MCDmcm(r, s)**



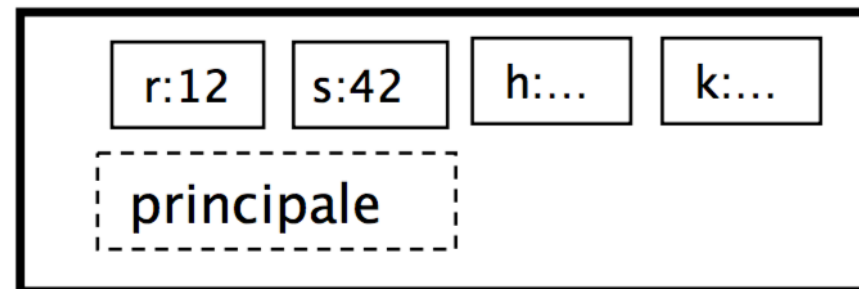
(1)

# Funzioni

$[h,k] = \text{MCDmcm}(r, s)$



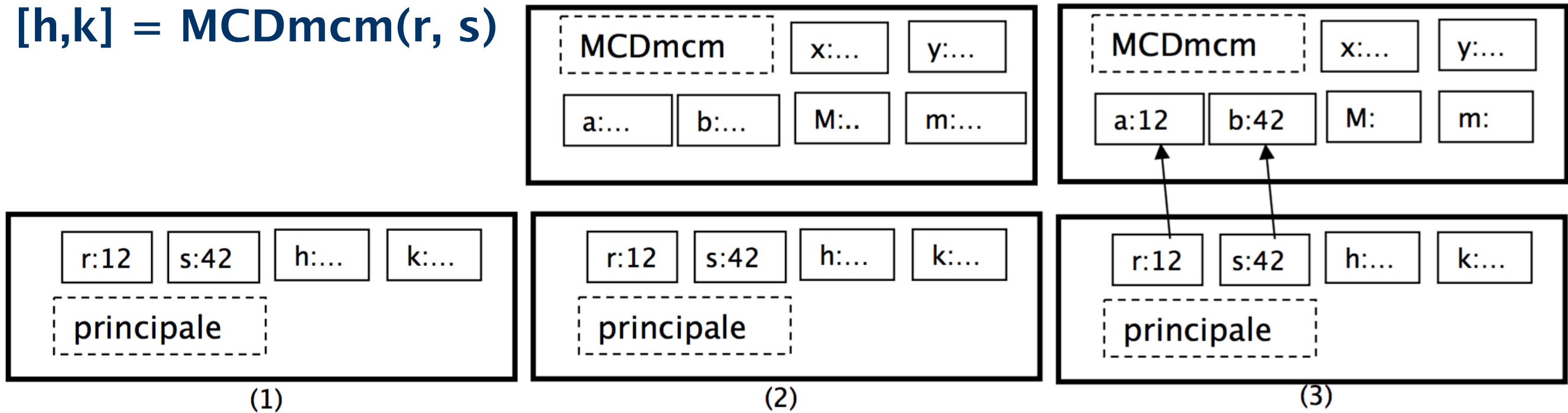
(1)



(2)

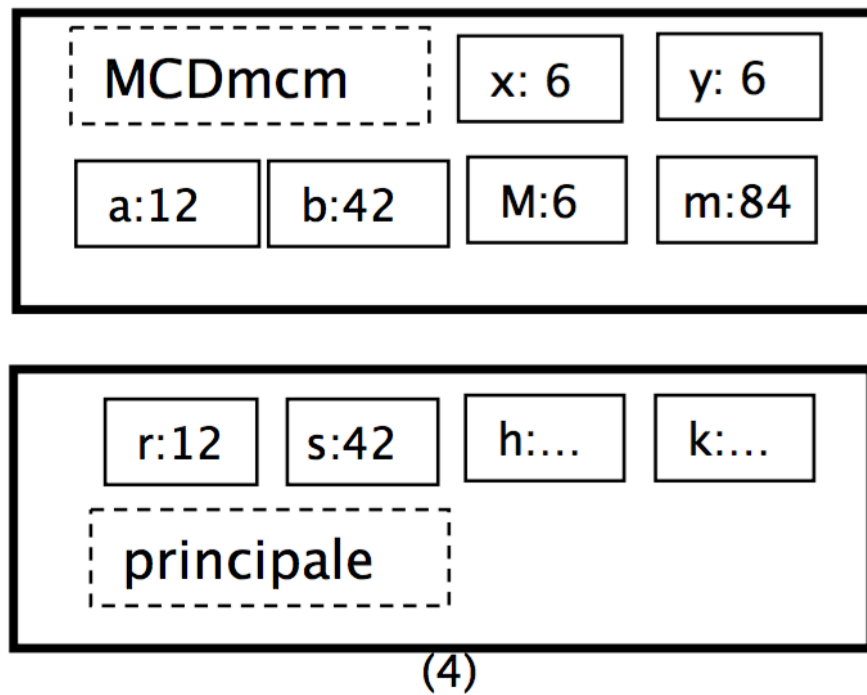
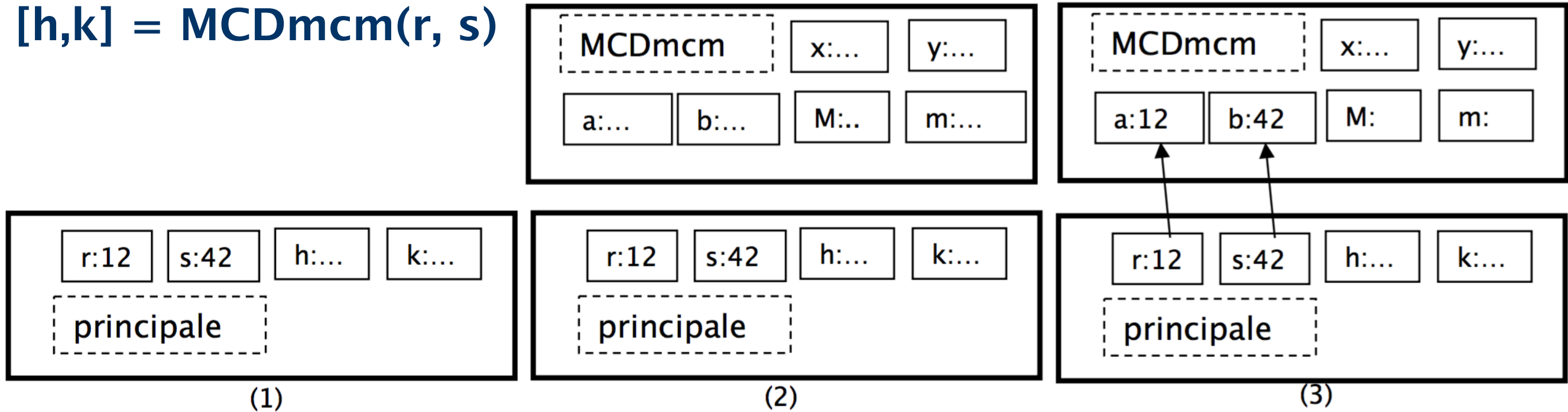
# Funzioni

$[h,k] = \text{MCDmcm}(r, s)$



# Funzioni

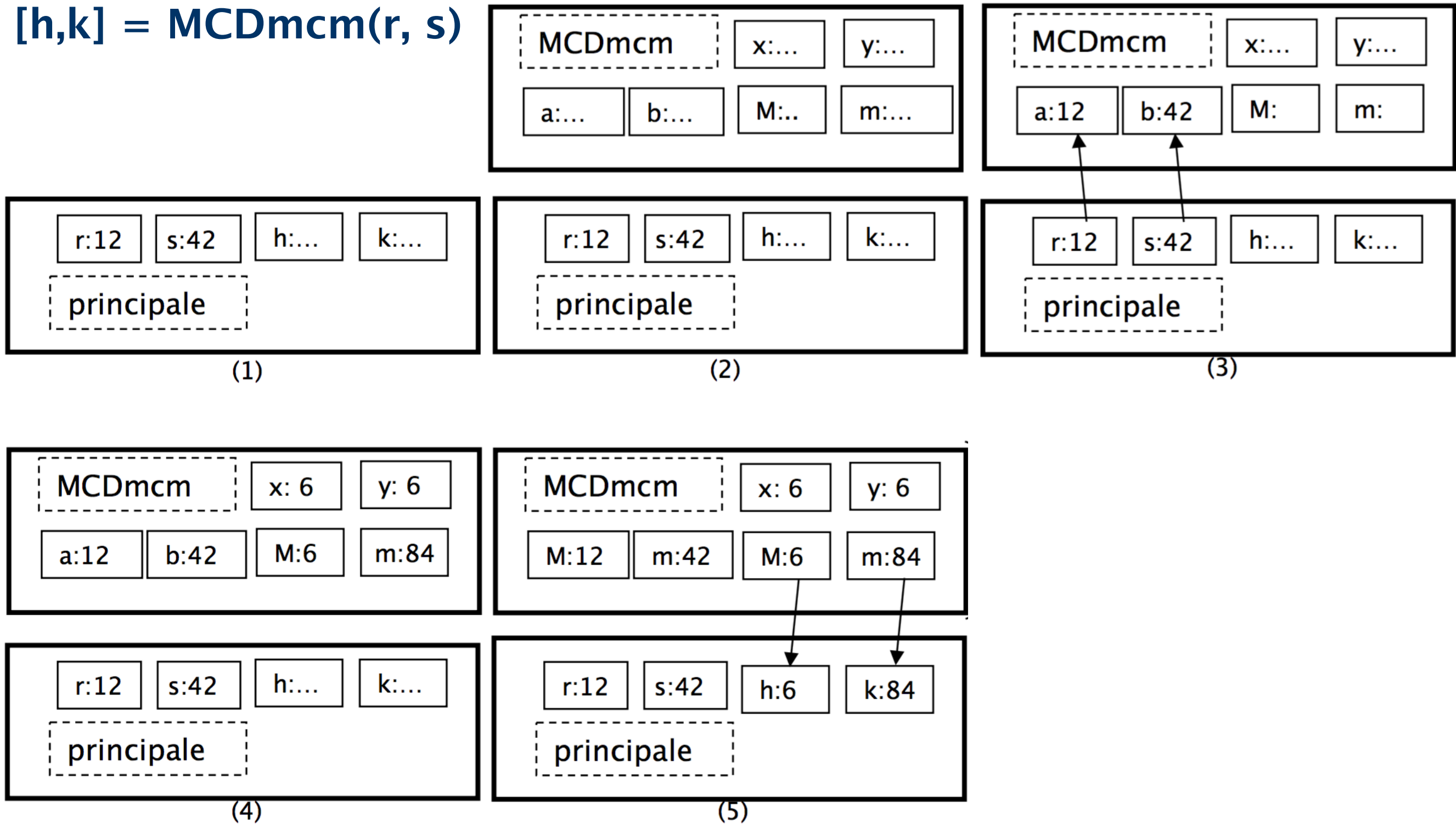
$[h,k] = \text{MCDmcm}(r, s)$





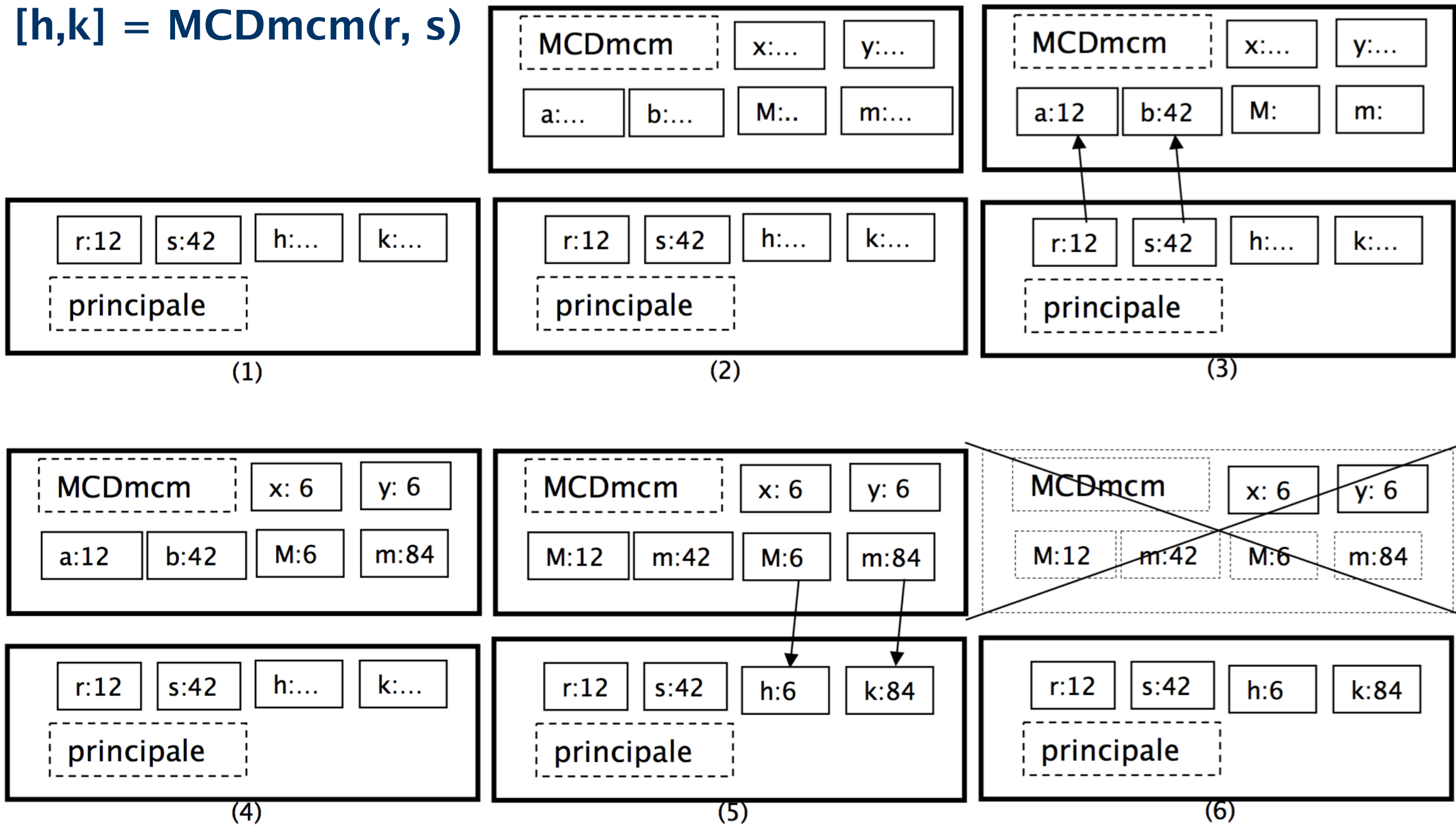
# Funzioni

$[h,k] = \text{MCDmcm}(r, s)$



# Funzioni

$[h,k] = \text{MCDmcm}(r, s)$



# 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

---

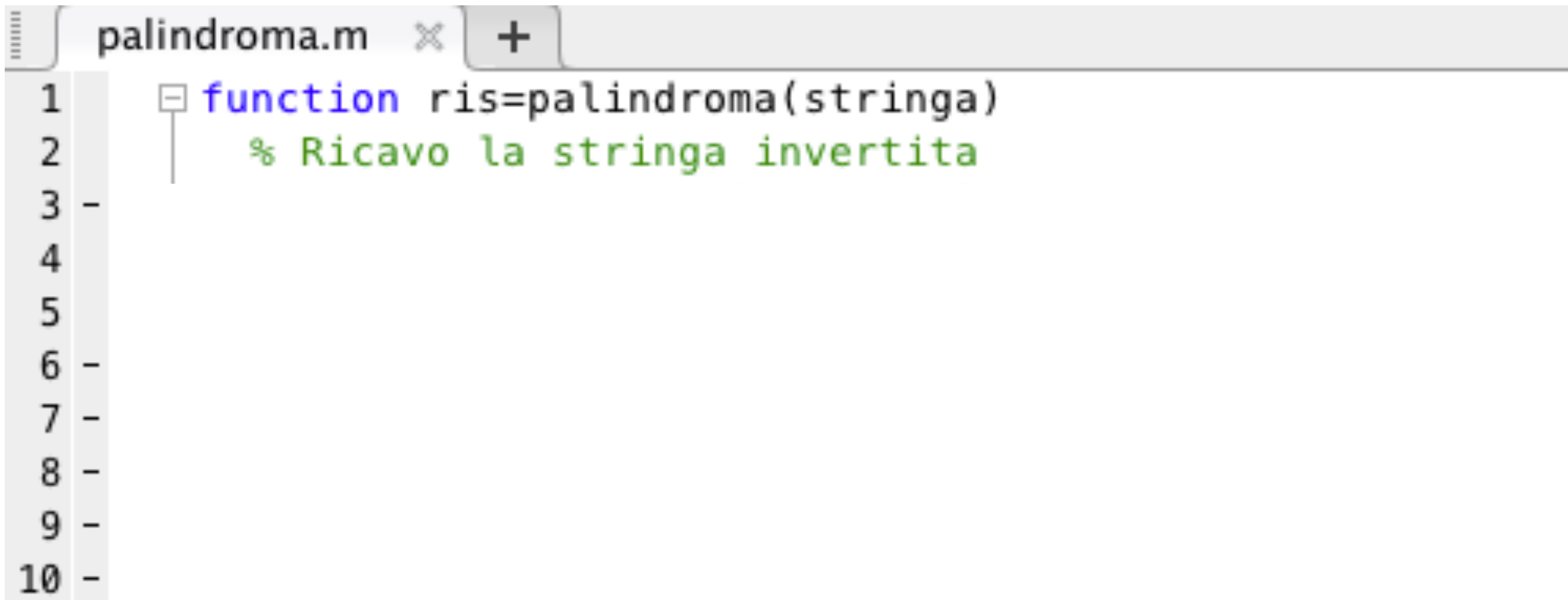


The image shows a MATLAB editor window with the file name 'palindroma.m'. The left margin displays line numbers from 1 to 10. The main editing area is currently blank.

Line Number	Content
1	
2	
3	-
4	
5	
6	-
7	-
8	-
9	-
10	-

# Esercizio: Funzioni e stringhe

---



The image shows a MATLAB editor window with a single tab titled 'palindroma.m'. The code is as follows:

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

The code defines a function named 'palindroma' that takes a string 'stringa' as input and returns a value 'ris'. A comment on line 2 indicates the goal is to find the reversed string. The editor interface includes a line number column on the left and a toolbar at the top with a close button and a plus sign for additional tabs.

# 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
```

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') Vettori (ripasso)~~

~~(20') Funzioni e stringhe~~

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

(10') Pausa

(20') Maschere di bit (ordinamento v2.0)

(20') Funzioni e divisori

(30') 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.

# Agenda

---

~~(20') Vettori (ripasso)~~

~~(20') Funzioni e stringhe~~

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

(10') Pausa

(20') Maschere di bit (ordinamento v2.0)

(20') Funzioni e divisori

(30') Struct (film)


# Esercizio: Maschere di bit (ordinamento v2)

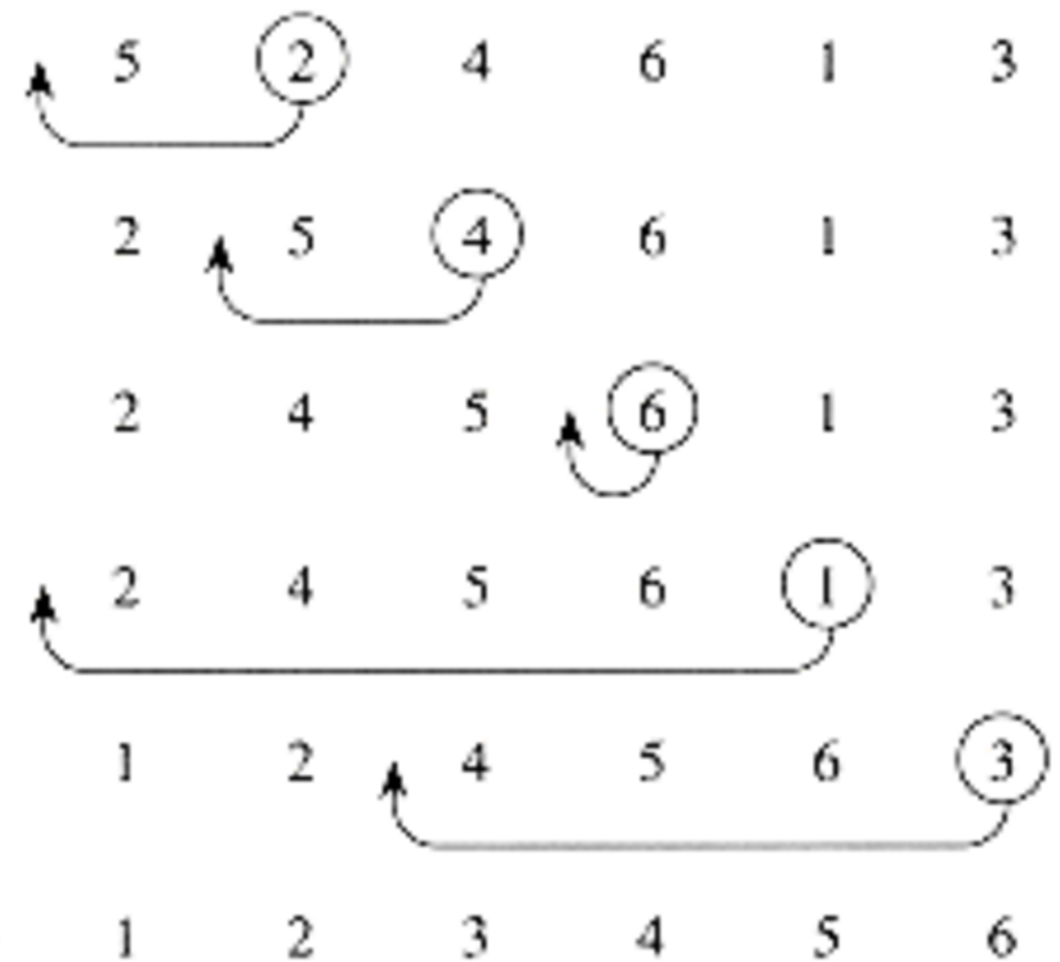
- Si ordini un array di n elementi facendo uso delle istruzioni messe a disposizione da matlab.

*Algoritmo di ordinamento: "Insertion sort"*  
([https://it.wikipedia.org/wiki/Insertion\\_sort](https://it.wikipedia.org/wiki/Insertion_sort))




# Esercizio: Maschere di bit (ordinamento v2)

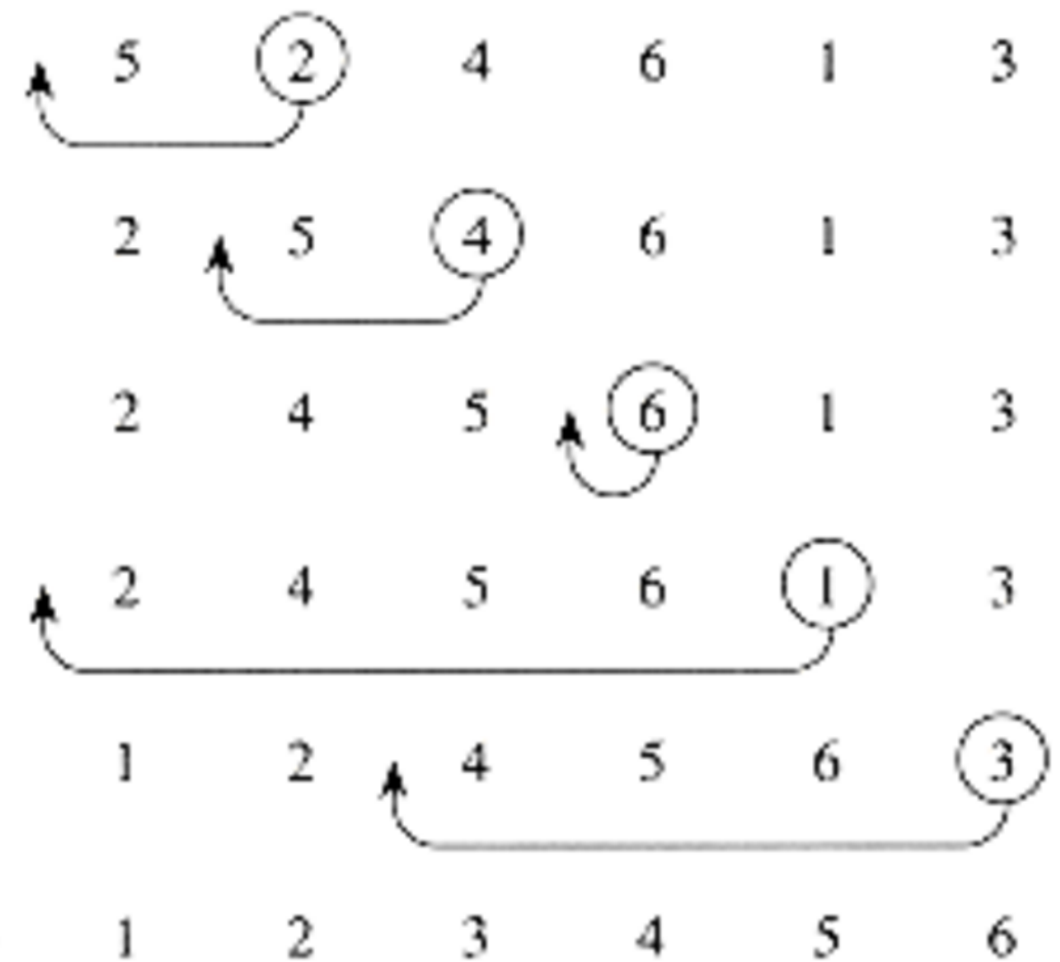
a  [5 2 4 6 1 3]




# Esercizio: Maschere di bit (ordinamento v2)

a  [5 2 4 6 1 3]

a\_ord = a(1);

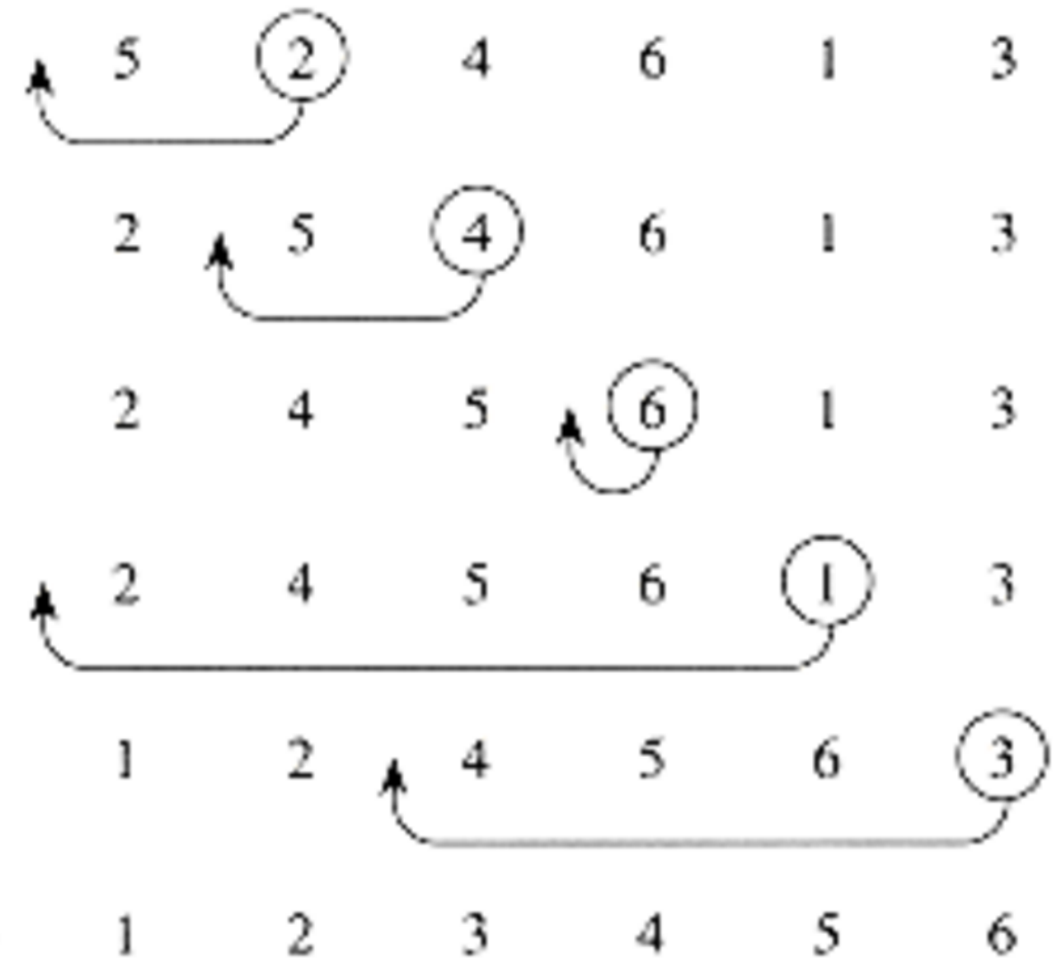


# Esercizio: Maschere di bit (ordinamento v2)

a  [5 2 4 6 1 3]

```
a_ord = a(1);
```

```
for i=2:length(a)
```

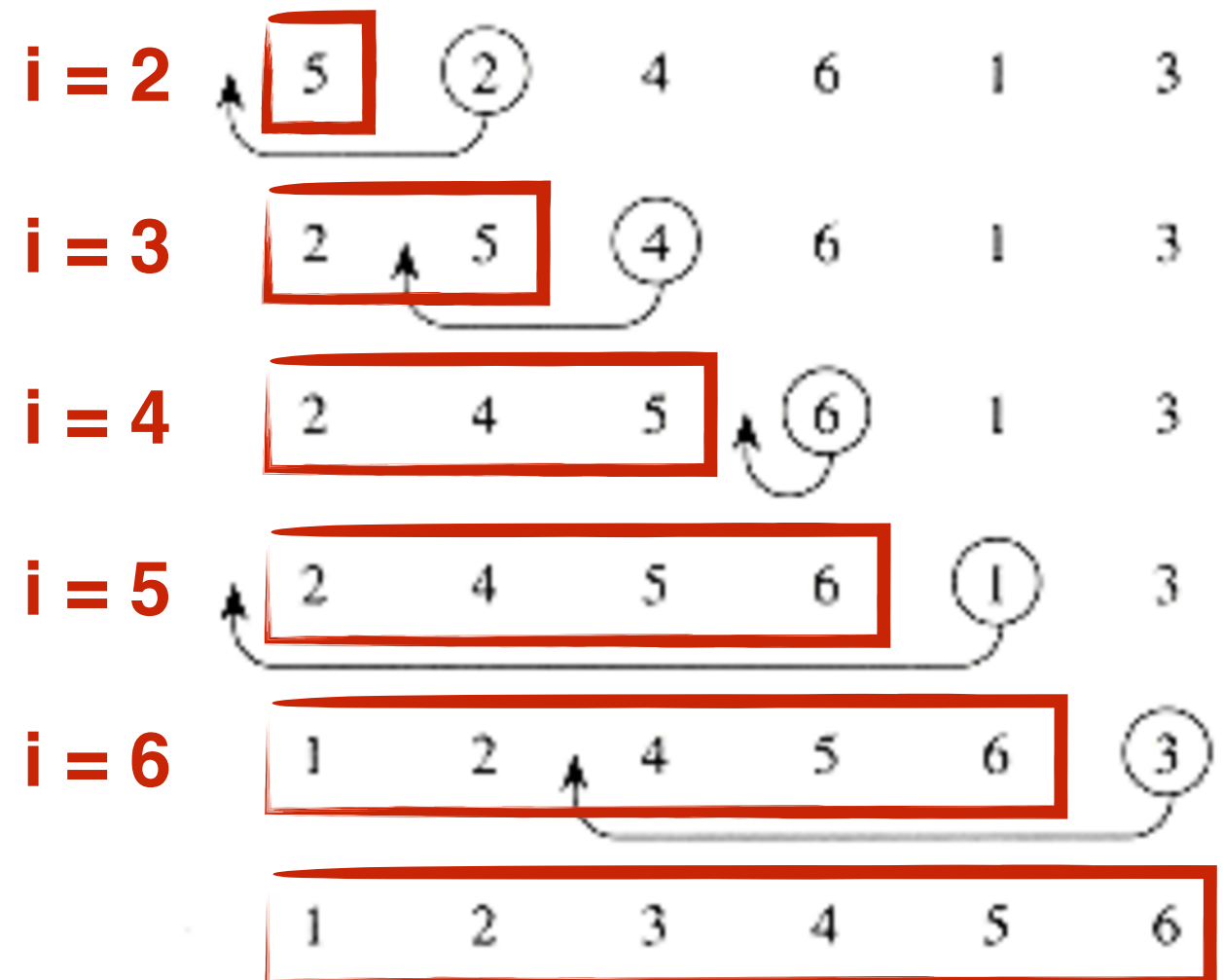


# Esercizio: Maschere di bit (ordinamento v2)

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 v2)

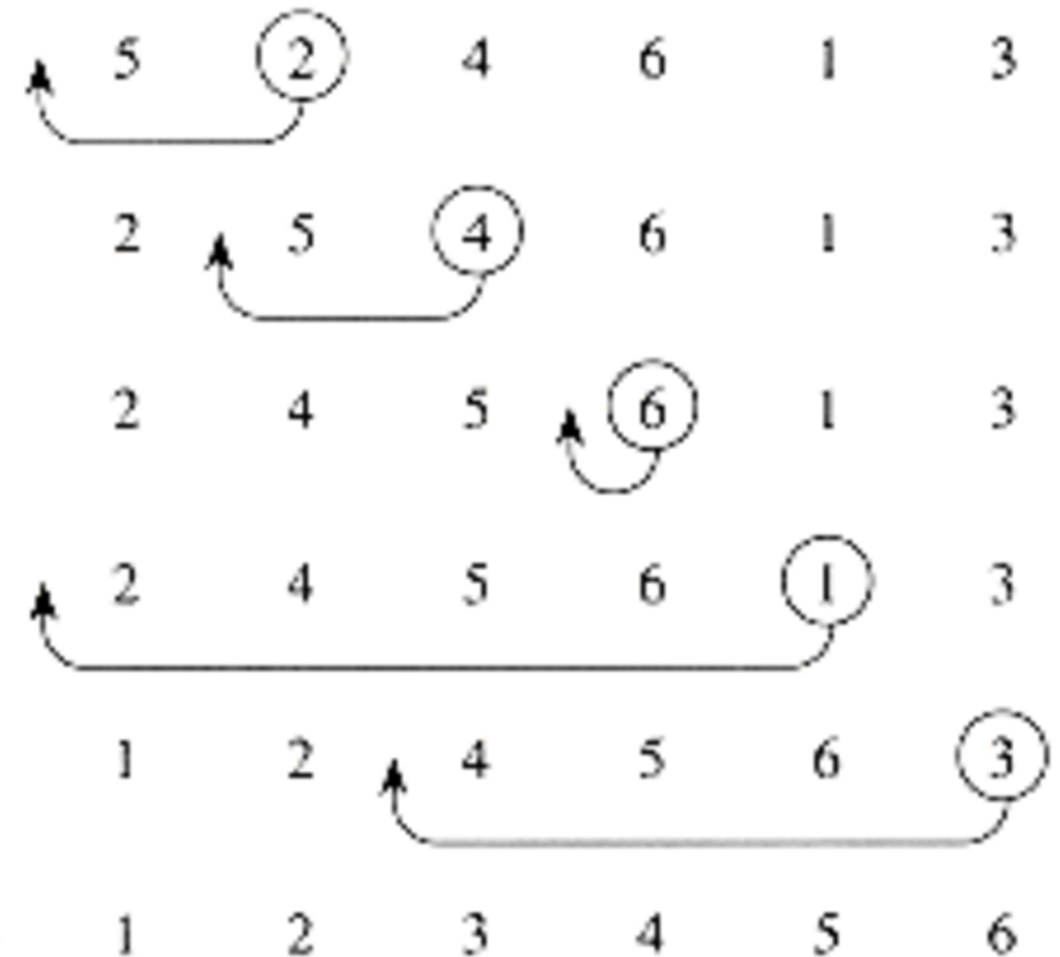
a ≡ [5 2 4 6 1 3]

```
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)];
```





# Ripasso: maschera, selezione, concatenazione

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

**maschera**

Vettore di **bit**:

[0 0 **1** 0 ... 0 **1 1 1** 0 0]

**1** se:  $v(i) \leq e$   
0 altrimenti

Vettore di **bit**:

[**1 1** 0 **1** ... **1** 0 0 0 **1 1**]

**1** se:  $v(i) > e$   
0 altrimenti

# Ripasso: maschera, **selezione**, concatenazione

---

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

**selezione**



**v**([0 0 **1** 0 ... 0 **1 1 1** 0 0])    **v**([**1 1** 0 **1** ... **1** 0 0 0 **1 1**])

Crea un **vettore**  
che contiene gli **elementi** di **v**  
**solo** nelle **posizioni degli 1**

# Ripasso: maschera, selezione, **concatenazione**

---

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

**concatenazione**

[ array 1 array 2 array 3 ]

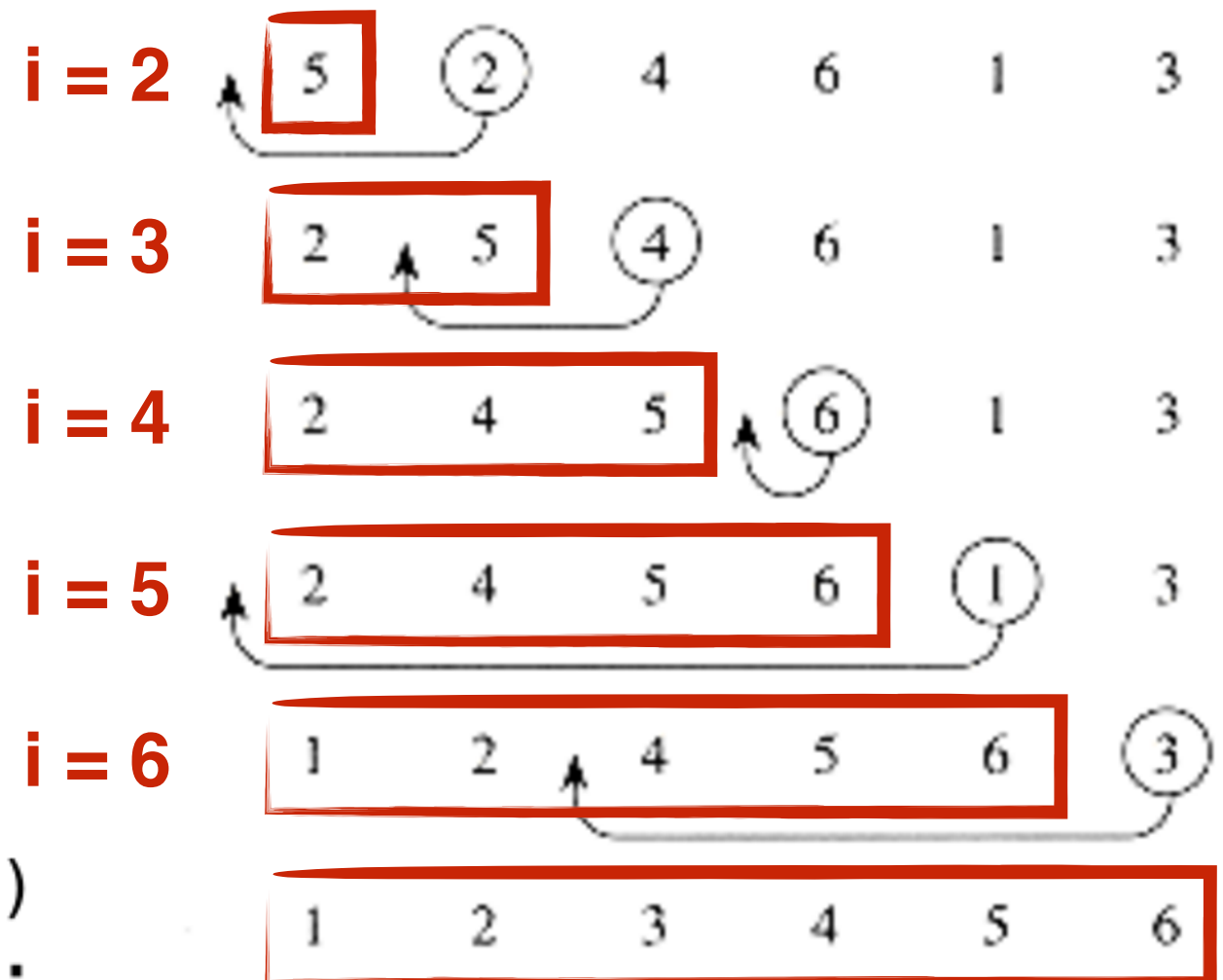
Crea un **vettore**  
che contiene gli **elementi** dei **vettori**  
**nell'ordine** specificato

# Esercizio: Maschere di bit (ordinamento v2)

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)];
```

# Agenda

---

~~(20') Vettori (ripasso)~~

~~(20') Funzioni e stringhe~~

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

~~(10') Pausa~~

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

(20') Funzioni e divisori

(30') Struct (film)

# Esercizio: Funzioni e divisori

---

Scrivere una funzione che riceva in ingresso un numero **N** e restituisca un array contenente **tutti i suoi divisori** (ad eccezione di 1 ed N stesso).

Nel caso N non abbia divisori, la funzione deve restituire l'array vuoto.

# Esercizio: Funzioni e divisori

---

```
N = input ( 'Inserire un numero: ' );
```

# Esercizio: Funzioni e divisori

---

```
N = input ( 'Inserire un numero: ' );
```

```
divisori(N)
```



# Esercizio: Funzioni e divisori

---

```
N = input ( 'Inserire un numero: ' );
```

```
mat2str(divisori(N))
```

# Esercizio: Funzioni e divisori

---

```
N = input ( 'Inserire un numero: ' );  
  
disp ( [ 'I divisori di ' num2str(N) ' sono ' mat2str(divisori(N)) ] );
```

# Esercizio: Funzioni e divisori

---

```
N = input ( 'Inserire un numero: ' );
```

```
disp ( [ 'I divisori di ' num2str(N) ' sono ' mat2str(divisori(N)) ] );
```

```
function v = divisori(N)
```

```
end
```

# Esercizio: Funzioni e divisori

---

```
N = input ( 'Inserire un numero: ' );
```

```
disp ( [ 'I divisori di ' num2str(N) ' sono ' mat2str(divisori(N)) ] );
```

```
function v = divisori(N)
```

```
%Possibili divisori
```

```
candidati = 2:N-1;
```

```
end
```

# Esercizio: Funzioni e divisori

---

```
N = input ( 'Inserire un numero: ' );  
  
disp ( [ 'I divisori di ' num2str(N) ' sono ' mat2str(divisori(N)) ] );
```

```
function v = divisori(N)  
%Possibili divisori  
candidati = 2:N-1;  
  
%Calcoliamo i divisori  
idx = mod(N,candidati)==0;
```

```
end
```

# Esercizio: Funzioni e divisori

---

```
N = input ( 'Inserire un numero: ' );

disp ( [ 'I divisori di ' num2str(N) ' sono ' mat2str(divisori(N)) ] );


function v = divisori(N)
%Possibili divisori
candidati = 2:N-1;

%Calcoliamo i divisori
idx = mod(N,candidati)==0;

    v = candidati(idx);

end
```

# Esercizio: Funzioni e divisori

---

```
N = input ( 'Inserire un numero: ' );

disp ( [ 'I divisori di ' num2str(N) ' sono ' mat2str(divisori(N)) ] );


function v = divisori(N)
%Possibili divisori
candidati = 2:N-1;

%Calcoliamo i divisori
idx = mod(N,candidati)==0;

% se ci sono divisori elimino i candidati superflui
if any(idx)
    v = candidati(idx);
else
    v=[];
end
```

# Agenda

---

~~(20') Vettori (ripasso)~~

~~(20') Funzioni e stringhe~~

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

~~(10') Pausa~~

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

~~(20') Funzioni e divisori~~

(30') Struct (film)



# 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

---

~~(20') Vettori (ripasso)~~

~~(20') Funzioni e stringhe~~

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

~~(10') Pausa~~

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

~~(20') Funzioni e divisori~~

~~(30') Struct (film)~~