

# Algoritmi su array

- Quando si usano gli array, si eseguono frequentemente alcune operazioni “tipiche”:
  - inizializzazione
  - lettura
  - stampa
  - ricerca del minimo e del massimo
  - ricerca di un valore
  - eliminazione di un valore
  - inserimento di un valore
  - ordinamento del vettore

## Ricerca di un valore nell'array

Si voglia verificare se in un array è presente almeno un'istanza di un certo valore. In caso positivo si fornisca l'indice dell'elemento.

- Quali costrutti usare ?
- Come gestire l'esito della ricerca ?

[Soluzione 1](#)

[Soluzione 2](#)

[Soluzione 3](#)

```
function main
% lettura di un array e ricerca dell'occorrenza di un elemento
% in un array
% Soluzione 1: ricerca esaustiva con FOR

% variabili utilizzate
% v: array in input
% i: indice per scorrere gli elementi dell'array
% n: dimensione dell'array fornito in input
% val: variabile contenente il valore da cercare
% pos: variabile contenente la posizione del valore nell'array


% input dimensione
n=input('Numero elementi: ');

% dimensionamento array
v=zeros(n,1);

% ciclo di lettura
for i=1:n
    fprintf('Valore %d: ',i);
    v(i)=input("");
end

% input valore da cercare
val=input('\nValore da cercare: ');

% ricerca dell'occorrenza
pos=0;

for i=1:n
    if(v(i)==val)
        pos=i;
    end
end
end
```

% stampa dei risultati

fprintf('\nArray letto:\n');

for i=1:n

    fprintf('V(%d): %g\n',i,v(i));

end

% si verifica se è stata effettivamente trovata un'occorrenza

if(pos==0)

    fprintf('\nNell"array non esistono occorrenze del valore %g\n',val);

else

    fprintf('\nIl valore %g si trova in posizione %d\n',val,pos);

end

```
function main
```

```
% lettura di un array e ricerca dell'occorrenza di un elemento
```

```
% in un array
```

```
% Soluzione 2: ricerca con WHILE, uso di flag booleano
```

```
% variabili utilizzate
```

```
% v:      array in input
```

```
% i:      indice per scorrere gli elementi dell'array
```

```
% n:      dimensione dell'array fornito in input
```

```
% val:    variabile contenente il valore da cercare
```

```
% pos:    variabile contenente la posizione del valore nell'array
```

```
% trovato: flag booleano che riporta l'esito della ricerca
```

```
% input dimensione
```

```
n=input('Numero elementi: ');
```

```
% dimensionamento array
```

```
v=zeros(n,1);
```

```
% ciclo di lettura
```

```
for i=1:n
```

```
    fprintf('Valore %d: ',i);
```

```
    v(i)=input("");
```

```
end
```

```
% input valore da cercare
```

```
val=input('\nValore da cercare: ');
```

% ricerca dell'occorrenza

pos=0;

trovato=0;

i=1;

while(i<=n & ~trovato)

if(v(i)==val)

pos=i;

trovato=1;

end

i=i+1;

end

% stampa dei risultati

fprintf('\nArray letto:\n');

for i=1:n

fprintf('V(%d): %g\n',i,v(i));

end

% si verifica se è stata effettivamente trovata un'occorrenza

if(~trovato)

fprintf('\nNell"array non esistono occorrenze del valore %g\n',val);

else

fprintf('\nIl valore %g si trova in posizione %d\n',val,pos);

end

```

function main
% lettura di un array e ricerca dell'occorrenza di un elemento
% in un array
% Soluzione 3: ricerca con WHILE, uso di pos nella condizione

% variabili utilizzate
% v:      array in input
% i:      indice per scorrere gli elementi dell'array
% n:      dimensione dell'array fornito in input
% val:    variabile contenente il valore da cercare
% pos:    variabile contenente la posizione del valore nell'array

% input dimensione
n=input('Numero elementi: ');

% dimensionamento array
v=zeros(n,1);

% ciclo di lettura
for i=1:n
    fprintf('Valore %d: ',i);
    v(i)=input("");
end

% input valore da cercare
val=input('\nValore da cercare: ');

% ricerca dell'occorrenza
pos=0;
i=1;

while(i<=n & pos==0)
    if(v(i)==val)
        pos=i;
    end
    i=i+1;
end

```

% stampa dei risultati

fprintf('\nArray letto:\n');

for i=1:n

    fprintf('V(%d): %g\n',i,v(i));

end

% si verifica se è stata effettivamente trovata un'occorrenza

if(pos==0)

    fprintf('\nNell"array non esistono occorrenze del valore %g\n',val);

else

    fprintf('\nIl valore %g si trova in posizione %d\n',val,pos);

end

## Conteggio numero di occorrenze di un valore nell'array

Si voglia verificare se in un array è presente almeno un certo valore. In caso positivo si fornisca il numero delle istanze presenti.

[Soluzione](#)

## Ricerca posizioni delle occorrenze di un valore nell'array

Si voglia verificare se in un array è presente un certo valore. In caso positivo si forniscano le posizioni delle istanze presenti.

[Soluzione](#)



```

function main
% lettura di un array e conteggio del numero di occorrenze
% di un elemento in un array

% variabili utilizzate
% v: array in input
% i: indice per scorrere gli elementi dell'array
% n: dimensione dell'array fornito in input
% val: variabile contenente il valore da cercare
% cont: variabile contenente il numero di occorrenze trovate

% input dimensione
n=input('Numero elementi: ');

% dimensionamento array
v=zeros(n,1);

% ciclo di lettura
for i=1:n
    fprintf('Valore %d: ',i);
    v(i)=input("");
end

% input valore da cercare
val=input('\nValore da cercare: ');

% ricerca dell'occorrenza
cont=0;

for i=1:n
    if(v(i)==val)
        cont=cont+1;
    end
end

% stampa dei risultati
fprintf('\nArray letto:\n');
for i=1:n
    fprintf('V(%d): %g\n',i,v(i));
end

% si verifica se è stata effettivamente trovata un'occorrenza
if(cont==0)
    fprintf('\nNell"array non esistono occorrenze del valore %g\n',val);
else
    fprintf('\nIl valore %g è presente %d volte nell"array\n',val,cont);
end

```

```

function main
% lettura di un array e ricerca delle posizioni delle occorrenze
% di un elemento in un array

% variabili utilizzate
% v: array in input
% i: indice per scorrere gli elementi dell'array
% n: dimensione dell'array fornito in input
% val: variabile contenente il valore da cercare
% pos: array contenente gli indici delle occorrenze
% cont: variabile contenente il numero di occorrenze trovate


% input dimensione
n=input('Numero elementi: ');

% dimensionamento array
v=zeros(n,1);
pos=zeros(n,1);

% ciclo di lettura
for i=1:n
    fprintf('Valore %d: ',i);
    v(i)=input("");
end

% input valore da cercare
val=input('\nValore da cercare: ');

% ricerca dell'occorrenza
cont=0;

for i=1:n
    if(v(i)==val)
        cont=cont+1;
        pos(cont)=i;
    end
end
end

```

% stampa dei risultati

fprintf('\nArray letto:\n');

for i=1:n

    fprintf('V(%d): %g\n',i,v(i));

end

% si verifica se è stata effettivamente trovata un'occorrenza

if(cont==0)

    fprintf('\nNell"array non esistono occorrenze del valore %g\n',val);

else

    fprintf('\nIl valore %g è presente %d volte nell"array\n',val,cont);

    fprintf('in queste posizioni: ');

    for i=1:cont

        fprintf('%d ',pos(i));

    end

    fprintf('\n');

end

## Ricerca posizioni delle occorrenze del minimo nell'array

Si voglia cercare il valore minimo in un array insieme con le posizioni delle varie occorrenze.

[Soluzione](#)

```
function main
```

```
% lettura di un array, ricerca del minimo e ricerca  
% delle posizioni delle occorrenze del minimo
```

```
% variabili utilizzate
```

```
% v: array in input
```

```
% i: indice per scorrere gli elementi dell'array
```

```
% n: dimensione dell'array fornito in input
```

```
% min: variabile contenente il valore da cercare
```

```
% pos: array contenente gli indici delle occorrenze
```

```
% cont: variabile contenente il numero di occorrenze trovate
```

```
% input dimensione
```

```
n=input('Numero elementi: ');
```

```
% dimensionamento array
```

```
v=zeros(n,1);
```

```
pos=zeros(n,1);
```

```
% ciclo di lettura
```

```
for i=1:n
```

```
    fprintf('Valore %d: ',i);
```

```
    v(i)=input("");
```

```
end
```

% ricerca del minimo

cont=1;

pos(1)=1;

min=v(1);

for i=2:n

if(v(i)<min)

% trovato nuovo minimo

% si reinizializza il minimo corrente

% ed il vettore delle occorrenze

cont=1;

pos(1)=i;

min=v(i);

elseif(v(i)==min)

% trovata nuova occorrenza del minimo corrente

% si aggiorna il vettore delle occorrenze

cont=cont+1;

pos(cont)=i;

end

end

% stampa dei risultati

fprintf('\nArray letto:\n');

for i=1:n

fprintf('V(%d): %g\n',i,v(i));

end

% stampa del minimo e delle occorrenze

fprintf('\nIl valore minimo nell"array è %g.\n',min);

fprintf('E" presente %d volte nelle seguenti posizioni: ',cont);

for i=1:cont

fprintf('%d ',pos(i));

end

fprintf('\n');

## Ricerca di un valore in un array ordinato

Si voglia verificare se in un array ordinato è presente almeno un'istanza di un certo valore. In caso positivo si fornisca l'indice dell'elemento.

- Quali costrutti usare ?
- Come gestire l'esito della ricerca ?
- Come sfruttare l'ordinamento dell'array ?

[Soluzione](#)

```
function main
```

```
% lettura di un array ordinato e ricerca dell'occorrenza  
% di un elemento
```

```
% variabili utilizzate
```

```
% v:    array in input
```

```
% i:    indice per scorrere gli elementi dell'array
```

```
% j:    indice per scorrere gli elementi dell'array
```

```
% n:    dimensione dell'array fornito in input
```

```
% val:   variabile contenente il valore da cercare
```

```
% pos:   variabile contenente la posizione del valore nell'array
```

```
% trovato: flag booleano che riporta l'esito della ricerca
```

```
% input dimensione
```

```
n=input('Numero elementi: ');
```

```
% dimensionamento array
```

```
v=zeros(n,1);
```

```
% ciclo di lettura
```

```
for i=1:n
```

```
    fprintf('Valore %d: ',i);
```

```
    v(i)=input("");
```

```
end
```

```
% input valore da cercare
```

```
val=input('\nValore da cercare: ');
```



% ricerca dell'occorrenza

pos=0;

trovato=0;

i=1;

while(i<=n & v(i)<=val & ~trovato)

if(v(i)==val)

pos=i;

trovato=1;

end

i=i+1;

end

% stampa dei risultati

fprintf('\nArray letto:\n');

for j=1:n

fprintf('V(%d): %g\n',i,v(j));

end

% si verifica se è stata effettivamente trovata un'occorrenza

if(~trovato)

fprintf('\nNell"array non esistono occorrenze del valore %g\n',val);

fprintf('Ultimo indice considerato: %d\n',i);

else

fprintf('\nIl valore %g si trova in posizione %d\n',val,pos);


end

## Ricerca di un valore in un array ordinato

- Quanti confronti si eseguono con l'algoritmo appena visto ?
- E' possibile organizzare opportunamente la ricerca per ridurre i confronti ?

## Ricerca di un valore in un array ordinato


- Si supponga di dover cercare il valore 15
- Si confronti inizialmente il valore con l'elemento centrale
- Siccome  $15 > 9$  possiamo trascurare la prima metà dell'array perché sicuramente non conterrà il valore cercato



1	1
2	3
3	4
4	6
5	9
6	10
7	13
8	14
9	18

## Ricerca di un valore in un array ordinato


- Concentriamo l'attenzione sulla parte superiore e nuovamente confrontiamo il valore con l'elemento centrale della parte "superstite" dell'array
- Siccome  $15 > 13$ , possiamo trascurare la metà inferiore della parte di array che stiamo considerando



1	1
2	3
3	4
4	6
5	9
6	10
7	13
8	14
9	18

## Ricerca di un valore in un array ordinato

- Anche questa volta l'elemento centrale della parte di array in esame è minore di 15, per cui si considera la parte di array successiva all'elemento centrale




1	1
2	3
3	4
4	6
5	9
6	10
7	13
8	14
9	18

## Ricerca di un valore in un array ordinato

- Questa volta la parte di array è costituita da un solo elemento, diverso dal valore cercato.
- Non ci sono più altre parti di array da esplorare e quindi la ricerca è finita con esito negativo.
- Quanti confronti abbiamo fatto in tutto ?

1	1
2	3
3	4
4	6
5	9
6	10
7	13
8	14
9	18



## Ricerca binaria

- L'algoritmo descritto si definisce di ricerca *binaria* o *dicotomica*.
- Come implementarlo ?
- Quali costrutti utilizzare ?

[Soluzione](#)

```
function main
```

```
% lettura di un array ordinato e ricerca dell'occorrenza  
% di un elemento
```

```
% variabili utilizzate
```

```
% v:      array in input
```

```
% n:      dimensione dell'array fornito in input
```

```
% val:    variabile contenente il valore da cercare
```

```
% posin:  variabile contenente la posizione dell'estremo inferiore della  
%         parte di array in esame
```

```
% posfin: variabile contenente la posizione dell'estremo inferiore della  
%         parte di array in esame
```

```
% posmed: variabile contenente la posizione dell'elemento centrale nella  
%         parte di array in esame
```

```
% trovato: flag booleano che riporta l'esito della ricerca
```

```
% input dimensione
```

```
n=input('Numero elementi: ');
```

```
% dimensionamento array
```

```
v=zeros(n,1);
```

```
% ciclo di lettura
```

```
for i=1:n
```

```
    fprintf('Valore %d: ',i);
```

```
    v(i)=input("");
```

```
end
```

```
% input valore da cercare
```

```
val=input('\nValore da cercare: ');
```

```
% ricerca dell'occorrenza
```

```
posin=1;
```

```
posfin=n;
```

```
trovato=0;
```

```
while(~trovato & posin<=posfin)
```

```
    % si determina l'elemento centrale
```

```
    posmed=floor((posin+posfin)/2);
```

```
    % si confronta con il valore da cercare
```

```
    if(v(posmed)>val)
```

```
        % si considera la metà inferiore
```

```
        posfin=posmed-1;
```

```
    elseif(v(posmed)<val)
```

```
        % si considera la metà superiore
```

```
        posin=posmed+1;
```

```
    else
```

```
        % trovato il valore
```

```
        trovato=1;
```

```
        pos=posmed;
```

```
    end
```

```
end
```

```
% stampa dei risultati
```

```
fprintf('\nArray letto:\n');
```

```
for j=1:n
```

```
    fprintf('V(%d): %g\n',j,v(j));
```

```
end
```

```
% si verifica se è stata effettivamente trovata un'occorrenza
```

```
if(~trovato)
```

```
    fprintf('\nNell"array non esistono occorrenze del valore %g\n',val);
```

```
else
```

```
    fprintf('\nIl valore %g si trova in posizione %d\n',val,pos);
```

```
end
```