Titolo unità didattica: Array e insiemi [08]

Titolo modulo : Algoritmi di base su insiemi - parte 1 [01-T]

Insiemi e array: unione e intersezione

Argomenti trattati:

- ✓ rappresentazione di insiemi come array
- ✓ algoritmo per l'unione di due insiemi
- ✓ algoritmo per l'intersezione di due insiemi

Prerequisiti richiesti: P1-07-03-T

problemi con insiemi
richiami:

$$A = \{2, 4, 6, 3\} \qquad B = \{4, 3, 1\}$$

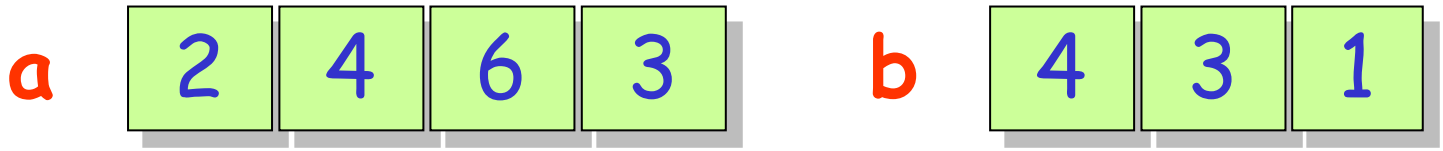
la **cardinalità** di un insieme è il numero di oggetti dell'insieme

$$c_A = 4 \qquad c_B = 3$$

in un insieme un oggetto **non** può apparire più volte

insiemi rappresentati mediante array

$A = \{2, 4, 6, 3\}$ $B = \{4, 3, 1\}$

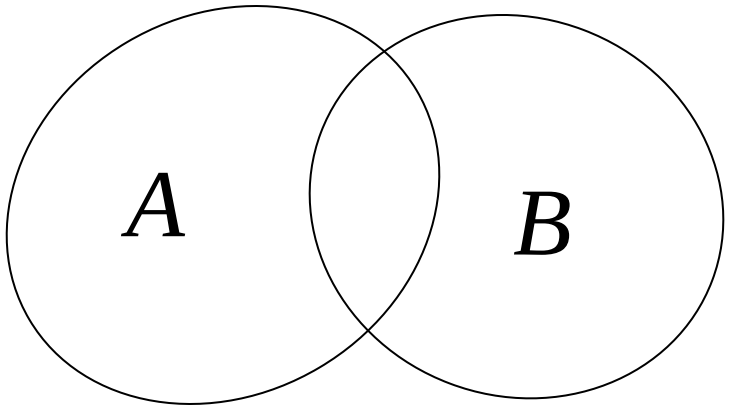


un insieme, i cui oggetti sono dati di tipo scalare, può essere rappresentato mediante un array 1D (di quel tipo) di size almeno uguale alla cardinalità dell'insieme

problemi con insiemi
unione di due insiemi:

$$A = \{2, 4, 6, 3\} \qquad B = \{4, 3, 1\}$$

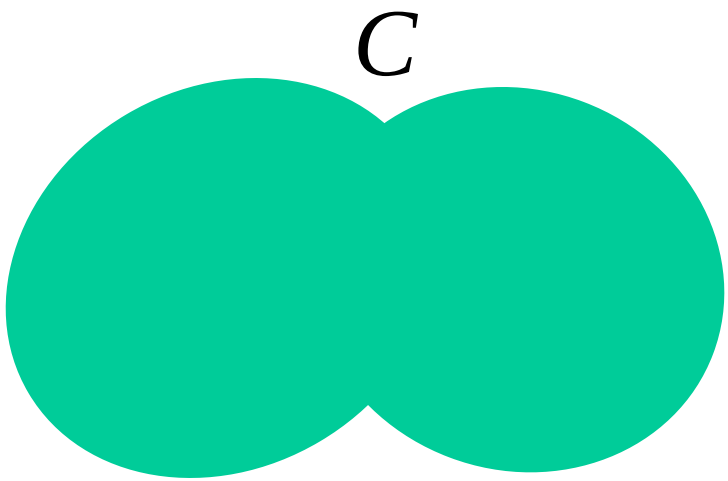
$$C = A \cup B = \{2, 4, 6, 3, 1\}$$



problemi con insiemi
unione di due insiemi:

$$A = \{2, 4, 6, 3\} \qquad B = \{4, 3, 1\}$$

$$C = A \cup B = \{2, 4, 6, 3, 1\}$$



problemi con insiemi
unione di due insiemi:

$$A = \{2, 4, 6, 3\} \qquad B = \{4, 3, 1\}$$

$$C = A \cup B = \{2, 4, 6, 3, 1\}$$

$$c_C = 5$$

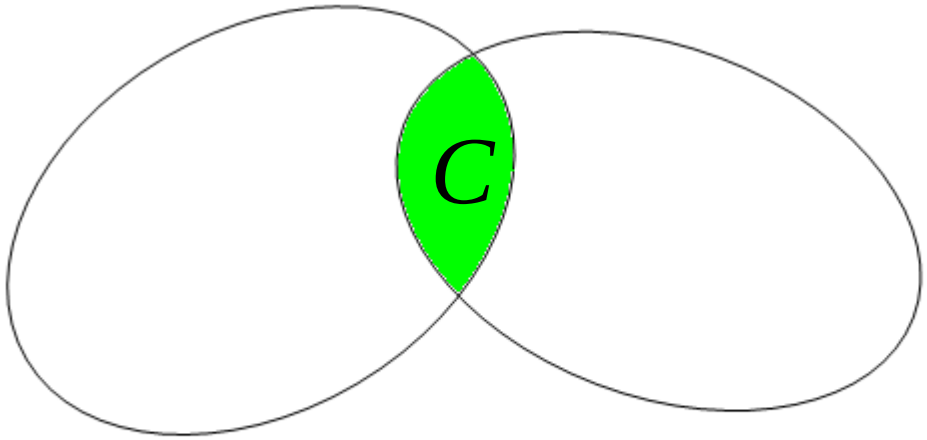
il numero di elementi dell'insieme unione è minore o al più uguale alla somma dei numeri degli elementi dei due insiemi

$$c_C \leq c_A + c_B$$

problemi con insiemi
intersezione di due insiemi:

$$A = \{2, 4, 6, 3\} \qquad B = \{4, 3, 1\}$$

$$C = A \cap B = \{4, 3\}$$



problemi con insiemi
intersezione di due insiemi:

$$A = \{2, 4, 6, 3\} \qquad B = \{4, 3, 1\}$$

$$C = A \cap B = \{4, 3\}$$

$$c_C = 2$$

il numero di elementi dell'insieme intersezione è minore o al più uguale al minimo tra i numeri degli elementi dei due insiemi

$$c_C \leq \min(c_A, c_B)$$

problemi con insiemi
unione di due insiemi:

dati di input: il primo insieme (variabile **a**), la sua cardinalità (variabile **n_a**), il secondo insieme (variabile **b**), la sua cardinalità (variabile **n_b**)
dati di output: l'insieme unione (variabile **c**), la sua cardinalità (variabile **n_c**)
costrutto ripetitivo: **for**
operazione ripetuta (al generico passo **i**):
verificare l'appartenenza di **b[i]** ad **a**
se **non** appartiene, deve essere un elemento di **c**

problemi con insiemi
unione di due insiemi:

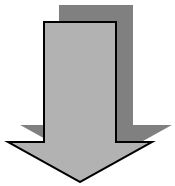
costrutto ripetitivo: for
operazione ripetuta (al generico passo **i**):
verificare l'appartenenza di **b[i]** ad **a**
se **non** appartiene, deve essere un elemento di **c**

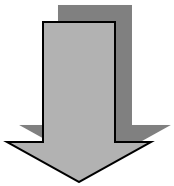
```
logical appartiene(int chiave, int array[], int n) ;
```

algoritmo di **ricerca sequenziale**
(costo n confronti al più)

unione

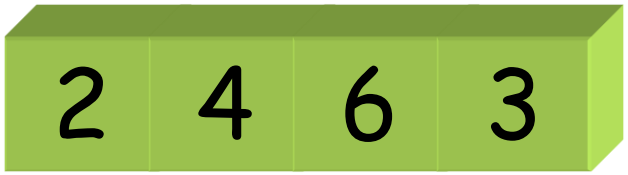
$A = \{2, 4, 6, 3\}$ $B = \{4, 3, 1\}$



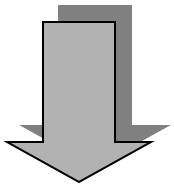


array **a**

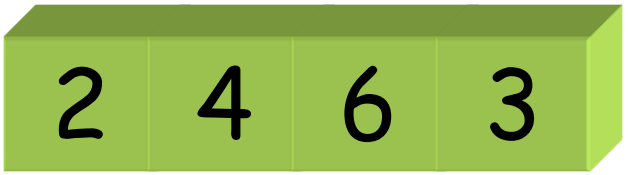
array **b**



$C = A \cup B = \{2, 4, 6, 3, 1\}$

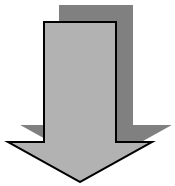


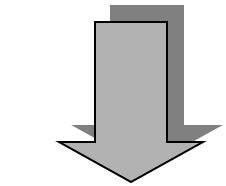
array **c**



unione

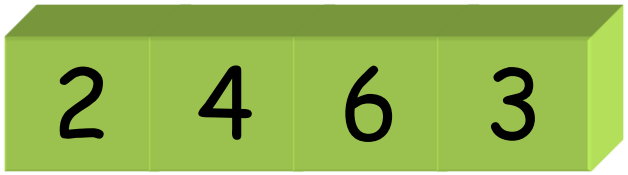
$A = \{2, 4, 6, 3\}$ $B = \{4, 3, 1\}$



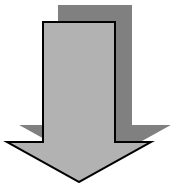


array **a**

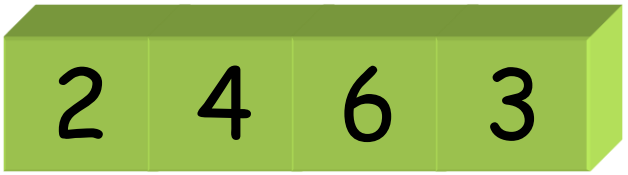
array **b**



$C = A \cup B = \{2, 4, 6, 3, 1\}$

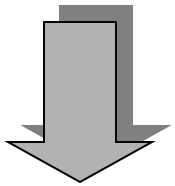


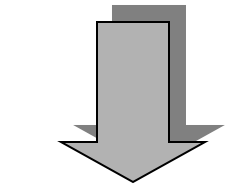
array **c**



unione

$A = \{2, 4, 6, 3\}$ $B = \{4, 3, 1\}$

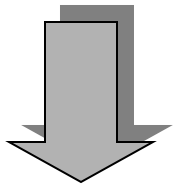




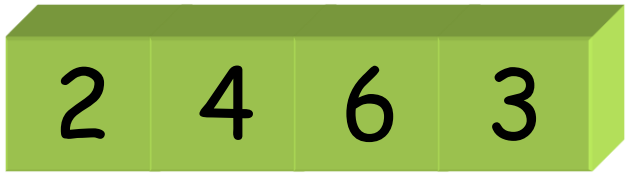
array **a**

array **b**

$C = A \cup B = \{2, 4, 6, 3, 1\}$

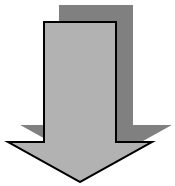


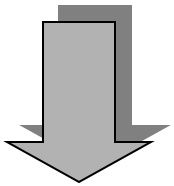
array **c**



unione

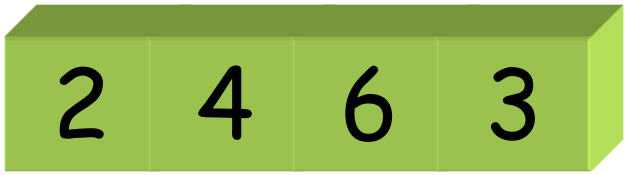
$A = \{2, 4, 6, 3\}$ $B = \{4, 3, 1\}$



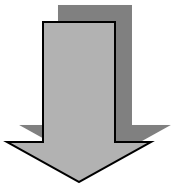


array **a**

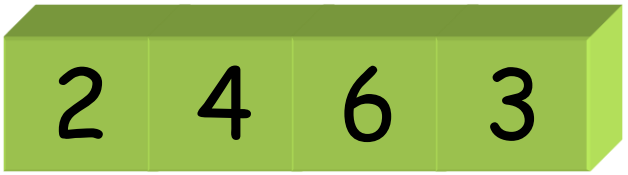
array **b**



$C = A \cup B = \{2, 4, 6, 3, 1\}$

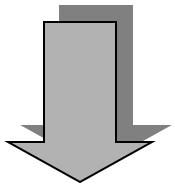


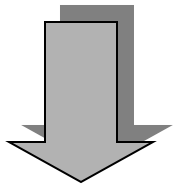
array **c**



unione

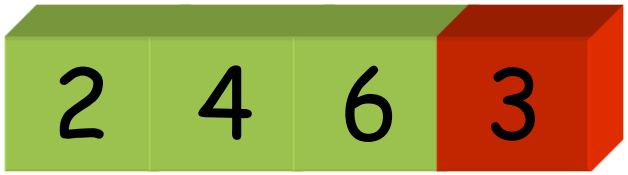
$A = \{2, 4, 6, 3\}$ $B = \{4, 3, 1\}$



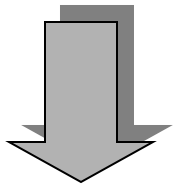


array **a**

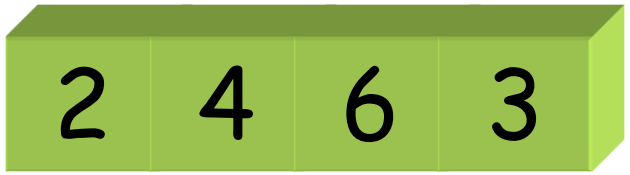
array **b**



$C = A \cup B = \{2, 4, 6, 3, 1\}$

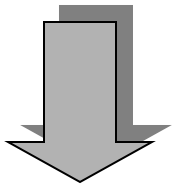


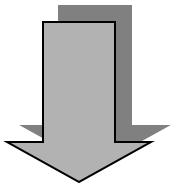
array **c**



unione

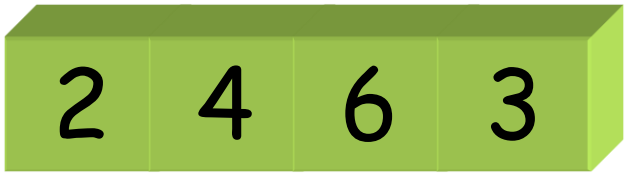
$A = \{2, 4, 6, 3\}$ $B = \{4, 3, 1\}$



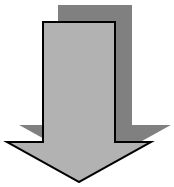


array **a**

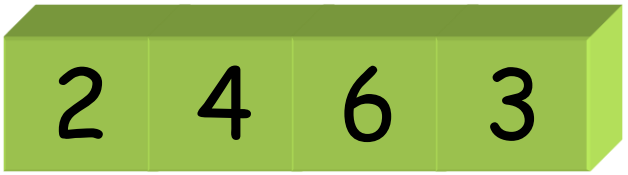
array **b**



$C = A \cup B = \{2, 4, 6, 3, 1\}$

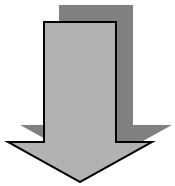


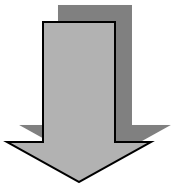
array **c**



unione

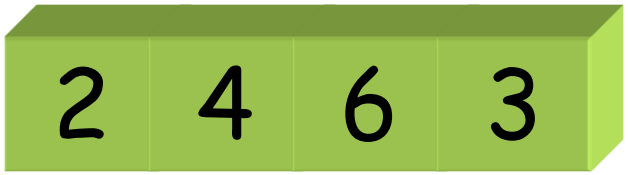
$A = \{2, 4, 6, 3\}$ $B = \{4, 3, 1\}$



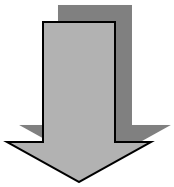


array **a**

array **b**



$C = A \cup B = \{2, 4, 6, 3, 1\}$



array **c**

```
void unione(in: float a[], int n_a, float b[], int
n_b; out: float c[], int n_c)
 int i;
/*  inizializzazione di c  */
  for (i=0; i < n_a; i++) {
     c[i] = a[i] ;
   }
  n_c = n_a ;
...
```

ATTENZIONE: da modificare in C

```
logical appartiene(float chiave,float array[],int n)
...
   for (i=0; i < n_b; i++) {
      if ( ! appartiene(b[i],a,n_a) )
          {
           c[n_c] = b[i] ;
           n_c = n_c+1 ;
          }
    }
}
```

n_b*n_a

confronti tra elementi degli array (al più)

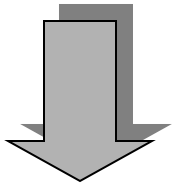
ATTENZIONE: da modificare in C

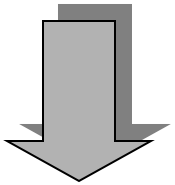
problemi con insiemi
intersezione di due insiemi:

dati di input: il primo insieme (variabile **a**), la sua cardinalità (variabile **n_a**), il secondo insieme (variabile **b**), la sua cardinalità (variabile **n_b**)
dati di output: l'insieme unione (variabile **c**), la sua cardinalità (variabile **n_c**)
costrutto ripetitivo: **for**
operazione ripetuta (al generico passo **i**):
verificare l'appartenenza di **a[i]** a **b**
se appartiene, deve essere un elemento di **c**

intersezione

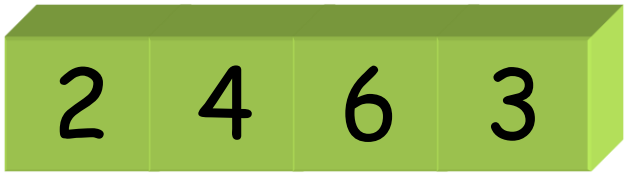
$A = \{2, 4, 6, 3\}$ $B = \{4, 3, 1\}$



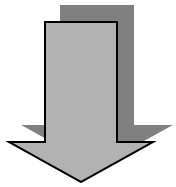


array **a**

array **b**



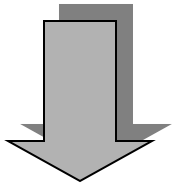
$C = A \cap B = \{4, 3\}$

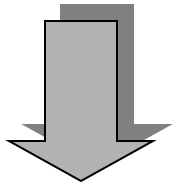


array **c**

intersezione

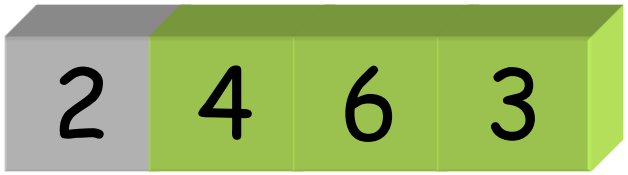
$A = \{2, 4, 6, 3\}$ $B = \{4, 3, 1\}$



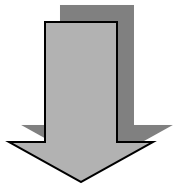


array **a**

array **b**



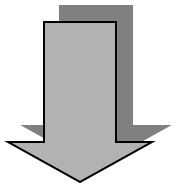
$C = A \cap B = \{4, 3\}$

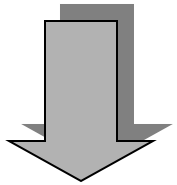


array **c**

intersezione

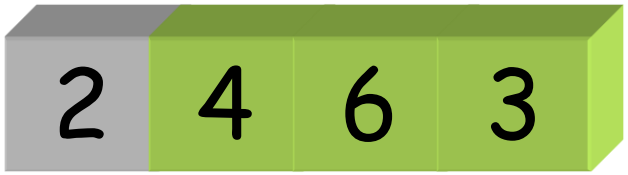
$A = \{2, 4, 6, 3\}$ $B = \{4, 3, 1\}$



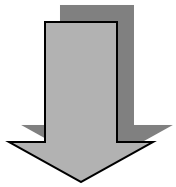


array **a**

array **b**



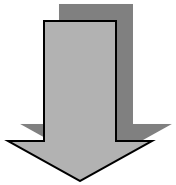
$C = A \cap B = \{4, 3\}$

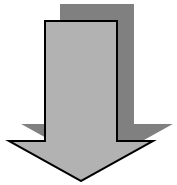


array **c**

intersezione

$A = \{2, 4, 6, 3\}$ $B = \{4, 3, 1\}$

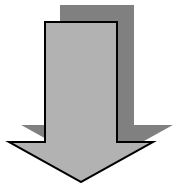




array **a**

array **b**

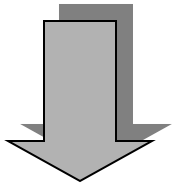
$C = A \cap B = \{4, 3\}$

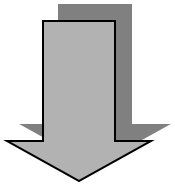


array **c**

intersezione

$A = \{2, 4, 6, 3\}$ $B = \{4, 3, 1\}$

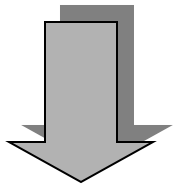




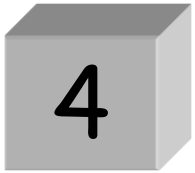
array **a**

array **b**

$C = A \cap B = \{4, 3\}$

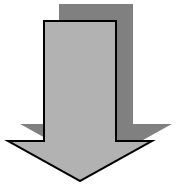


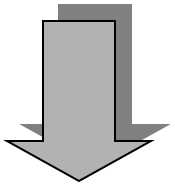
array **c**



intersezione

$A = \{2, 4, 6, 3\}$ $B = \{4, 3, 1\}$

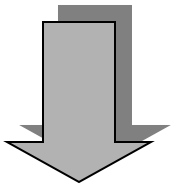




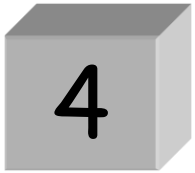
array **a**

array **b**

$C = A \cap B = \{4, 3\}$

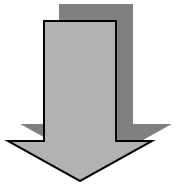


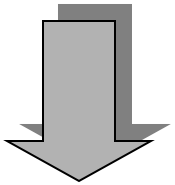
array **c**



intersezione

$A = \{2, 4, 6, 3\}$ $B = \{4, 3, 1\}$

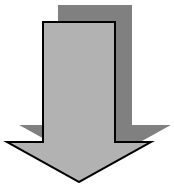




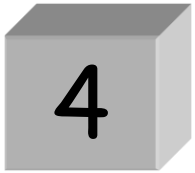
array **a**

array **b**

$C = A \cap B = \{4, 3\}$

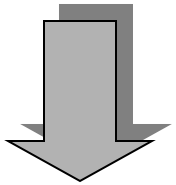


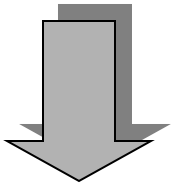
array **c**



intersezione

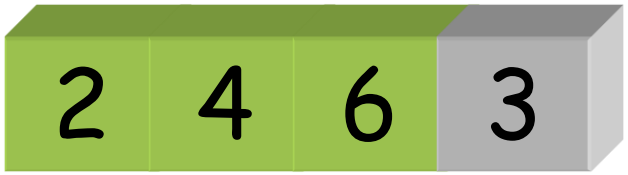
$A = \{2, 4, 6, 3\}$ $B = \{4, 3, 1\}$



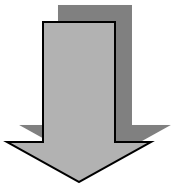


array **a**

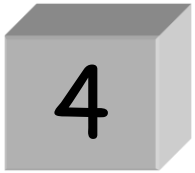
array **b**



$C = A \cap B = \{4, 3\}$

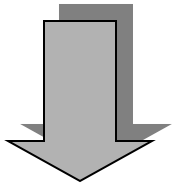


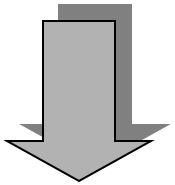
array **c**



intersezione

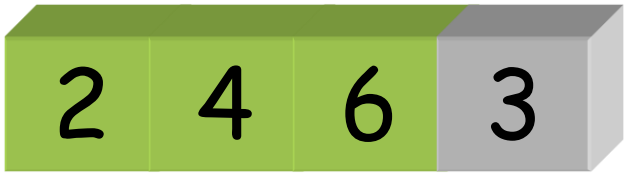
$A = \{2, 4, 6, 3\}$ $B = \{4, 3, 1\}$



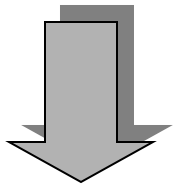


array **a**

array **b**



$C = A \cap B = \{4, 3\}$



array **c**

```
void intersezione( in: float a[], int n_a, float b[],
                   int n_b; out: float c[], int n_c) {
 int i ;
 n_c = 0 ;
 for (i=0; i < n_a; i++) {
      if (appartiene(a[i],b,n_b)
       {
        c[n_c] = a[i] ;
        n_c = n_c+1 ;
        }
  }
}
```

n_b*n_a
confronti tra elementi dei due array
(al più)

ATTENZIONE: da modificare in C

```
logical appartiene(float chiave,float array[],int n)
```