```
void azzera (int a[], int dim)
{ int i;
  for (i=0; i<dim; i++)
      a[i]=0;
}

main()
{ int a[5];
  azzera (a, 5);
  :
```

legge dall'esterno i valori dell'array

int *a

```
void lettura (int a[], int dim)
{ int i;
  for (i=0; i<dim; i++)
      scanf("%d", a[i]);
```

$l_1 + 0$ → $*l_1$

$*(a+i)$ → valore contenuto nella locazione l_1 = $*l_1$

$\&(*(l_1+i))$

SI ANNULLANO perché & è l'operazione inversa di *

```
void lettura (int a[], int dim)
{ int i;
  for (i=0; i<dim; i++)
      scanf ("%d", &a[i]);
}

main()
{ int a[5];
  int b[10];
  lettura (a, 5);
  lettura (b, 10);
  ...
```

dichiarazione di funzioni o/e procedure

uso

giovedì 27 settembre 2018 16:28

Scrivere una funzione che restituisca la somma di tutti gli elementi di un array di interi.

```
int somma (int a[], int dim)
{ int i;
  int S = 0;
  for (i=0; i<dim; i++)
      S = S + a[i];
  return s;
}
```

```
{int i;
 int s = a[0];
 for(i = 1; i<dim; i++)
     s = s + a[i]
 return s;
}
```

int s;

Scrivere una funzione che vale
vero se l'array a cui è
applicata è ordinato in senso
crescente
falso altrimenti.

giovedì 27 settembre 2018 16:36

boolean = {true, false}

↳ predefinito

(boolean x = true;)

boolean non è predefinito

in C al posto del tipo boolean
si usa il tipo int
con la convenzione che

0 corrisponde a falso
1 " " true

ORDINATO IN SENSO CRESCENTE

$\forall i \in [0, dim-1) . \; a[i+1] > a[i]$

1 2 5 7 crescente

1 2 2 7 non è crescente

$\forall i \in [0, dim-1) . \; a[i+1] \geq a[i]$

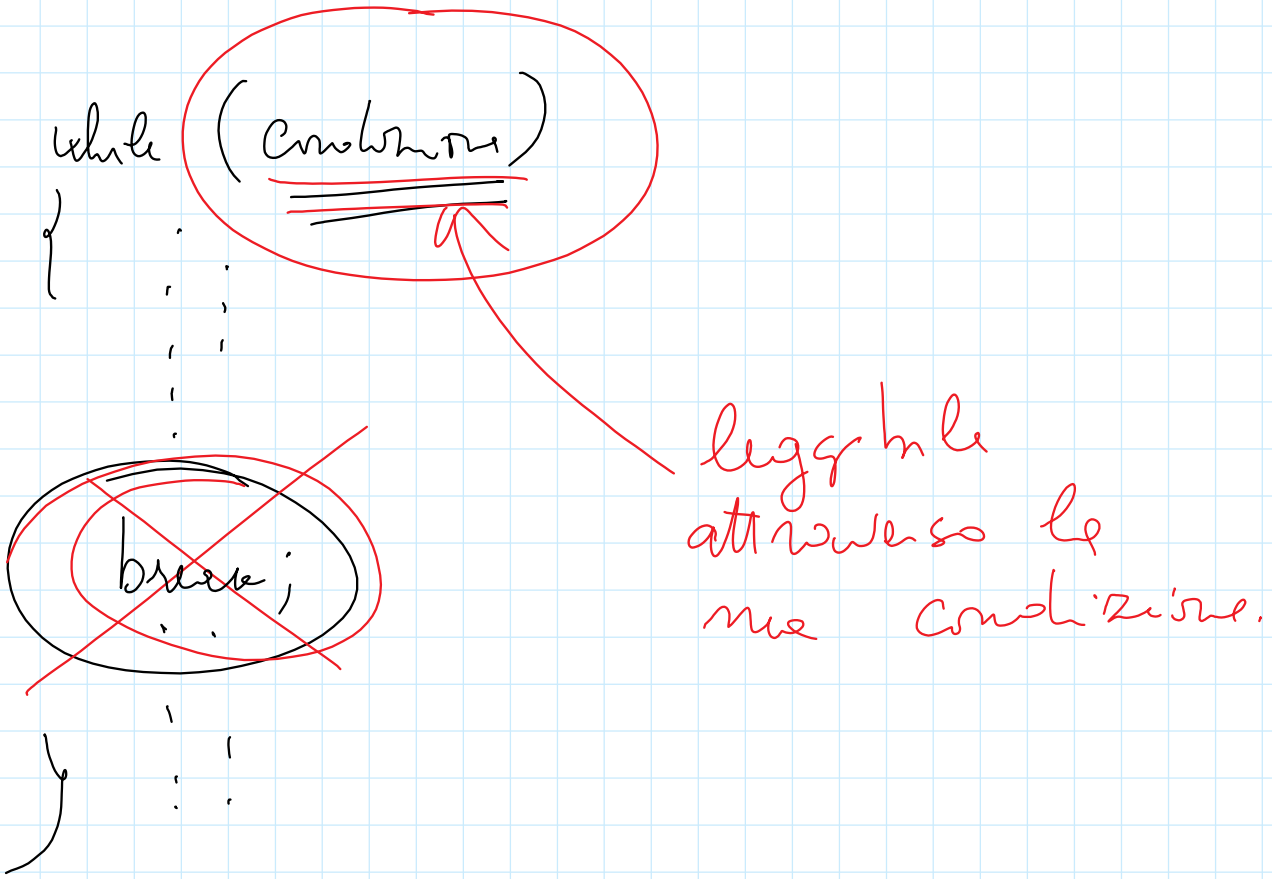
l'array è ordinato in senso

NON DECRESCENTE

giovedì 27 settembre 2018 16:44

rappresento boolean

```
int crescente (int a[], int dim)
{
```



```
int crescente (int a[], int dim)
{ int i;
  int ok = 1;
  for(i=0; i<dim-1; i++)
    if(a[i]>=a[i+1]) ok=0;
    else ok=1;
  return ok;
}
```

1 2 1 3 5

è inutile controllare gli elementi successivi ad una coppia "errata"

```
int crescente(int a[], int dim)
{ int i = 0;
  while (i < dim-1 && a[i] < a[i+1])
    i = i + 1;
  return i == dim - 1;
}
```

vero 1

falso 0

$i < dim - 1$

$i < 4 - 1$

$i < 3$

1	2	3	4
0	1	2	3

$i = 3$

false

$(i < dim - 1 \;\&\&\; a[i] < a[i+1])$

operatore &&

$i == dim - 1$

$!(i < dim - 1)$

$i >= dim - 1$

clamico: si valutano gli operandi se sono entrambi veri il risultato è "vero" (true), altrimenti "falso" (false)

Valutazione condizionale &&

A && B = if A then B
else false

Nella valutazione condizionale se A è falso non viene valutato il secondo operando.

(i < dim - 1 && a[i] < a[i+1])

in C la valutazione di && è condizionale

(a[i] < a[i+1] && i < dim - 1)

giovedì 27 settembre 2018 17:12

```
int crescente (int a[], int dim)
{ int i = 0;
  int ok = 1;
  while (i < dim-1 && ok)
    if (a[i] < a[i+1]) i = i+1;
    else ok = 0;
  return ok;
}
```

ok == 1 ≡ ok

non può mai dare errore

$i, i+1 \in [0, dim)$

$\forall$ dominio vuoto vale Vero

Scrivere una funzione che vale "Vero" se un elemento, dato come argomento, appartiene ad un array, "falso" altrimenti:

MEMBER

member 3 | 2 | 3 | 5 | 3 | = true (1)

" " | 2 | 1 | 5 | 1 | = falso (0)

```
void lettura(...) {...}
int member(int el, int a[], int dim)
{ int i=0;
  int ok=0;
  while(i<dim && !ok)
    if(a[i]==el) ok=1;
    else i=i+1;
  return ok;
}

main()
{ int a[100];
```

ok == 0 → not ~ → !

!A	se A è falso (in C se A==0)
≡	
¬A	!A è vero

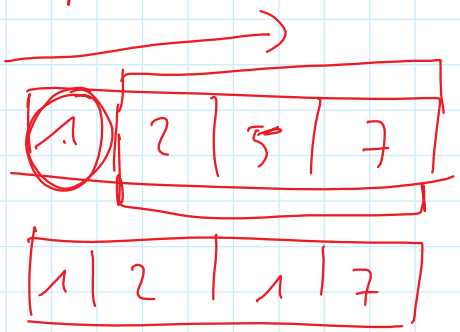
return OK;
}

main()
{ int a[100];
int x;
leggine(a,100);
scanf("%d", &x);
printf("%d", member(x,a,100));

!A	se A è falso (in C se A==0)
≡	
¬A	!A è vero

giovedì 27 settembre 2018 17:32

Scrivere una funzione che vale vero se tutti gli elementi di un array sono diversi tra loro (distinti)



true → 1

false → ∅

Ricerca di un elemento in una porzione di array (es: da un indice "inizio" a un indice "fine" escluso.

giovedì 27 settembre 2018 17:36

```
int member (int el, int a[],
            int inizio, int fine)
{ int i = inizio;
  int ok = 0;
  while (i < fine && !ok)
      if (a[i] == el) ok = 1;
      else i = i + 1;
  return ok;
}

main()
{ int a[100];
  int x;
  lettura (a, 100);
  scanf ("%d", &x);
```