Comando di iterazione determinato

for

```
for (i=0; i<N; i=i+1) Comando
```

In C non è vero che il for termina sempre

```
for (i=0; i<3; i=i+1) i=i-1;
```

(i++)

non termina.

Per chiarezza useremo, nei programmi,
- il for, quando sappiamo quante volte il comando dovrà essere iterato
- il while quando non lo sappiamo.

Scrivere una procedura che azzera tutti gli elementi di un array di interi.

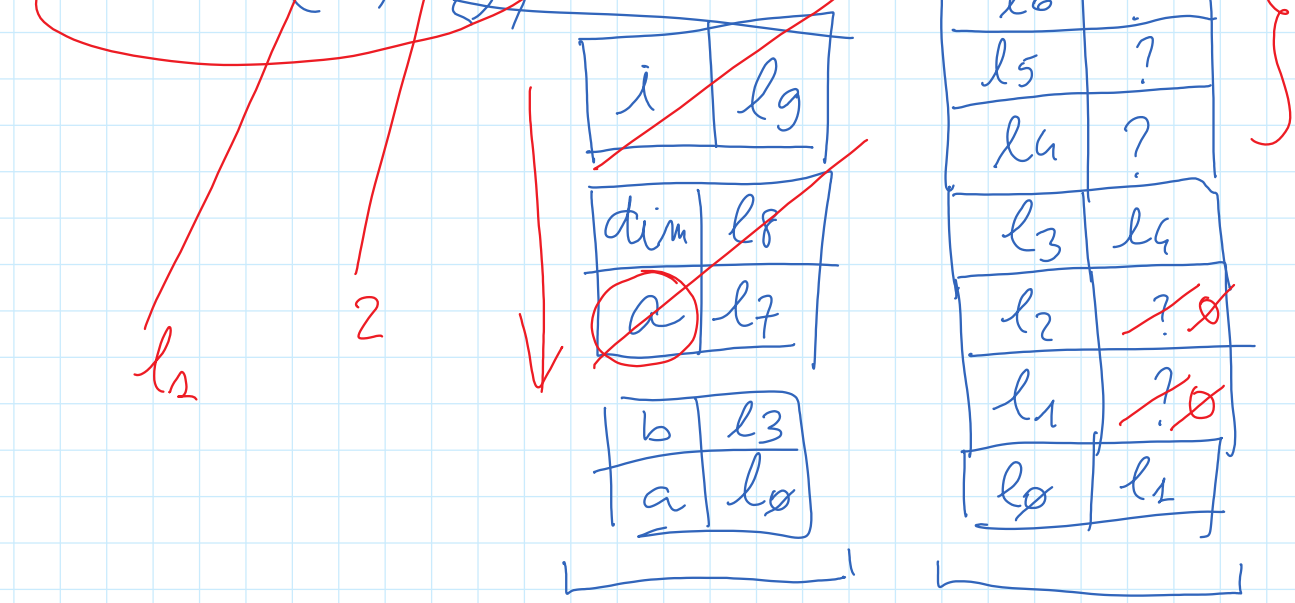
```
void azzera (int a[], int dim)
{ int i;
  for (i=0; i<dim; i++) a[i]=0;
}

main()
{ int a[2];
  int b[3];
  azzera (a, 2);
  azzera (b, 3);
}
```

int *a;

*(a+i)=0;

non le vedete



Procedura che legge gli elementi di un array.

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

main ()
{ int a[5];
  int b[10];
  read (a, 5);
  read (b, 10);
     :
```

```
main()
{ int s1, s2;
  int a[5];
  int b[10];
  read(a, 5);
  read(b, 10);
  s1 = sum(a, 5);
  s2 = sum(b, 10);
       :
```

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

Controllare che gli elementi di un array letto siano ordinati in modo crescente oppure no.

a

-2	3	5	7	8	25

ordinato in modo crescente

$$\left(\forall i \in [0, dim-1) \;.\; a[i] < a[i+1]\right)$$

(dove deve valere)

quantificazione universale con dominio $([0, dim))$

$a[dim]$ non esiste

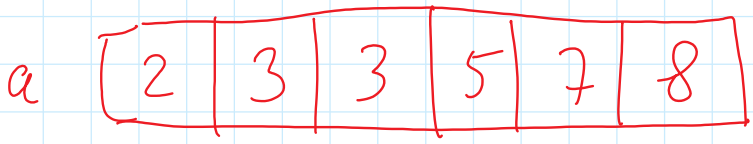
l'ultimo elemento di a ha indice $(dim-1)$

a | -2 | 3 | -1 | 5 | -6 | -3 | non è ordinato

ordinato in modo decrescente

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

a | 10 | 2 | -1 | -3 | -4 | -25 | in modo decrescente.



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

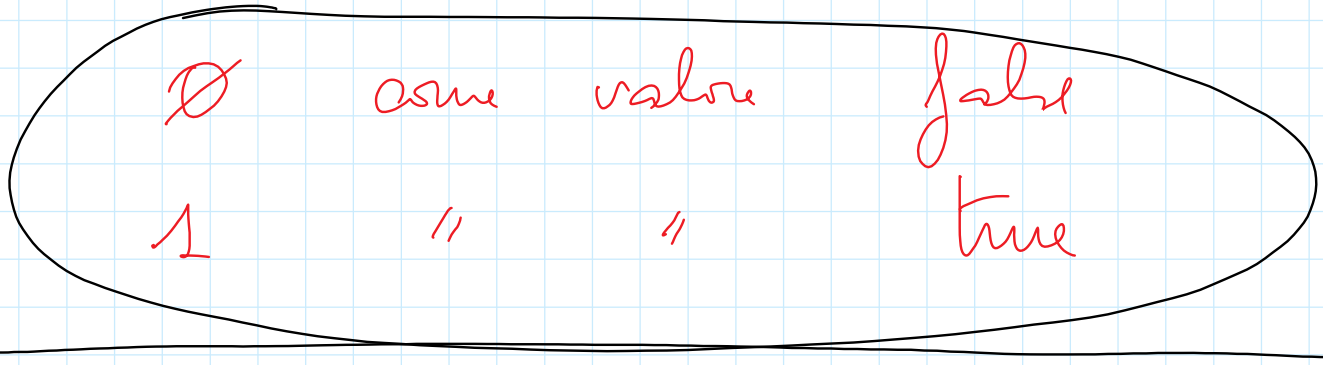
è ordinato in modo NON DECRESCENTE

Funzione che mi dice se l'array passato come argomento è in ordine crescente o no.

boolean = { true, false }

in C il tipo boolean non è predefinito!

In C ci siamo i valori interi



In realtà in C

∅ sta per false

tutti gli altri valori interi

stanno per true

definire le funzione ordinata

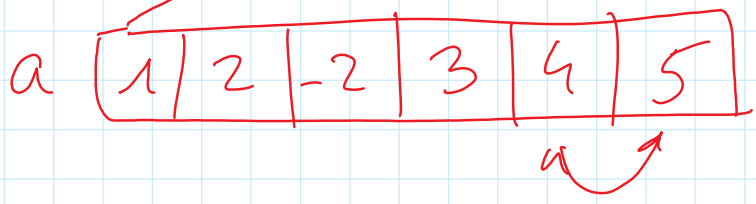
```
main ()
{ int ord;
  int a[5];
  int b[10];
  read (a, 5);
  read (b, 10);
  ord = ordinata (a, 5);
  ... = ordinata (b, 10);
```

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

Corretto

```
for (i=0; i< dim-1 ; i++)
     if (a[i] >= a[i+1]) tutto_ok = 0;
     else tutto_ok = 1;
```

non è corretto



Sostituire if for con

```
for (i=1; i<dim; i++)
    if (a[i-1] >= a[i]) tutto_ok = 0;
```

Se incontro una coppia "fuori posto" mi fermo immediatamente.

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

and logico

→ return (i == dim-1);

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

↳ and

$A \wedge B$ è vero se entrambi A e B sono veri

Si valutano A e B e si guarda se entrambi sono veri!!

$i = dim - 1$

$(\underbrace{dim - 1 < dim - 1}_{\text{false}} \;\&\&\; \underbrace{a[dim-1] < a[dim]}) \quad \leftarrow \text{errore non esiste}$

&& è un and condizionale

(cand)

A cand B = if A then B
else false

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

=

non viene valutata quando questo è falso

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

Se ma

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

uguale

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

tutto_ok == 1

(tutto_ok && i < dim-1)