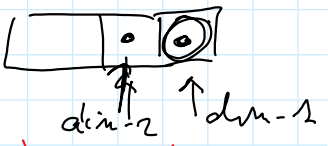
lunedì 2 ottobre 2017 08:51

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

=1 viene quando trovato è falso =0

i++;

```
int distinti (int v[], int dim)
{ int dist = 1;
  int i = 0;
  while (i < dim - 1 && dist)
    if (member (v[i], v, i+1, dim))
      dist = 0;
    else i++;
  return dist;
}
```



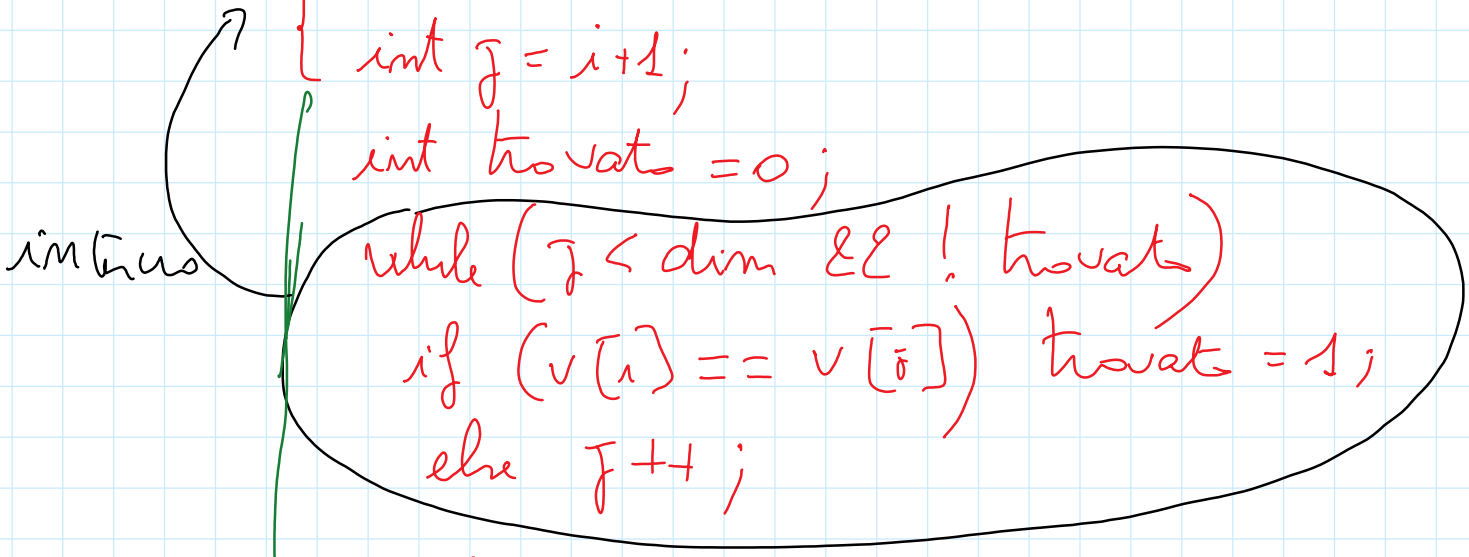
dist == 1

controllare se l'elemento a[i] compare tra i successivi, con indici [i+1, dim)

incluso ↑ escluso

lunedì 2 ottobre 2017 08:51

```
int dist (int v[], int dim)
{ int dist = 1;
  int i = 0;
  while (i < dim-1 && dist)
  { int j = i+1;
    int trovato = 0;
    while (j < dim && !trovato)
      if (v[i] == v[j]) trovato = 1;
      else j++;
    if (trovato) dist = 0;
    else i++;
  }
  return dist;
}
```



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

```
int member ( . . . . . )
{
  for (int i = inizio; i < fine; i++)
      (a[i] == el) return 1;
  return 0;
}
```

mai return nei cicli
(nelle iterazioni sia for
sia while)

```
int member ( . . . . )
{ int trovato = 0;
  for (i = inizio; i < fine && !trovato; i++)
    if (a[i] == el) trovato = 1;
  return trovato;
```

abbiamo usato un for con un while

```
}  return hovat;
```

```
int membro ( - - -  - - - - )
{
  for (int i = inizio ; i < fine ; i++)
    if (a[i] == el) i = fine;
  if (.....)
```

??

in un for non si deve modificare il valore della variabile di controllo (punto!)

lunedì 2 ottobre 2017 08:51

Scrivere una funzione che controlla se, in un array, esiste un elemento minore del successivo. Restituisce 1 se l'elemento esiste, 0 altrimenti.

a | 2 | -1 | $\Longrightarrow$ 0

a | 7 | 3 | 5 | -1 | $\Longrightarrow$ 1

a di dimensione dim

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

$(\exists i.\ i \in [0, dim-1) \wedge a[i] < a[i+1])$

$(\exists\, i \in [0, dim-1).\ F) \equiv (\exists i.\ i \in [0, dim-1) \wedge F)$

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

$\equiv$

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

$(\forall i \in [0, dim-1).\ F) \equiv (\forall i.\ i \in [0, dim-1) \Rightarrow F)$

esiste un elemento $a[i] < a[i+1]$

lunedì 2 ottobre 2017 08:51

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

lunedì 2 ottobre 2017 08:51

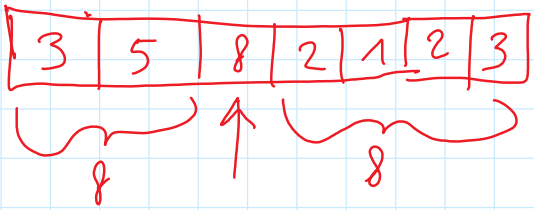
Tutti gli elementi di un array, escluso l'ultimo, sono più piccoli del successivo.

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

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

lunedì 2 ottobre 2017 08:51

Vogliamo controllare se in un array esiste un elemento con indice in $(0, dim-1)$ che è uguale alla somma di tutti gli elementi che lo precedono e uguale alla somma di tutti gli elementi che seguono.



$(0, dim-1)$

```
int somma(int a[], int inizio, int fine)
```

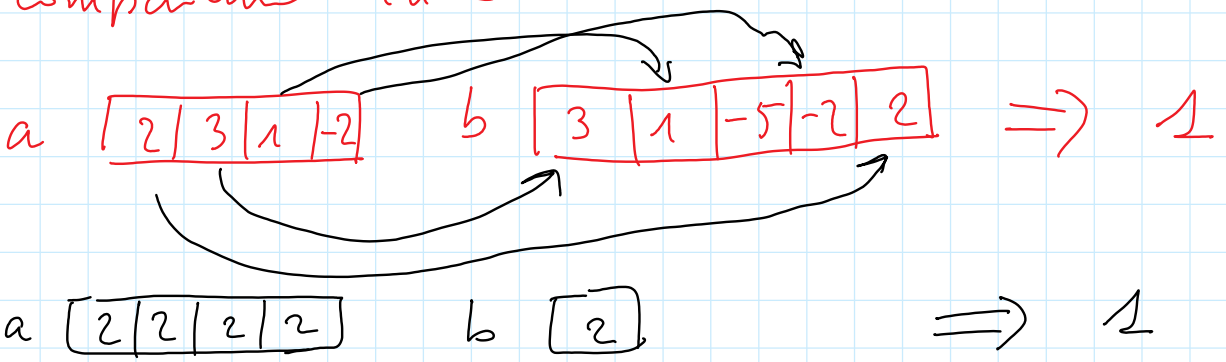
somma degli elementi di a con indici in [inizio, fine)

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

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

lunedì 2 ottobre 2017 08:51

Dati due array, a e b, di dimensione dima e dimb, rispettivamente, vogliamo controllare che tutti gli elementi di a compaiano in b




```
int compaiono (int a[], int dima, int b[],
               int dimb)
{
  :
```

$(\forall i \in [0, dima) .$
$\exists j \in [0, dimb) . \; a[i] = b[j])$

```
int member (int el, int v[], int dim)
{ .... }

int check (int a[], int dima, int b[], int dimb)
{
  int ok = 1;
  int i = 0;
  while (i < dima && ok)
    if (member (a[i], b, dimb)) i++;
    else ok = 0;
  return ok;
}
```

```
→ if (!member(a[i], b, dimb)) ok = 0;
  else i++;
```

lunedì 2 ottobre 2017 08:51

Scrivere una funzione che controlla che tutti gli elementi di un array siano distinti (tutti diversi)

Vale 1 se la proprietà è vera, 0 altrimenti

$(\forall i \in [0, dim) . \exists j . j \in [0, dim) \wedge j \neq i \wedge a[i] = a[j])$

dimensione dell'array a

$\equiv \forall i . i \in [0, dim) \Rightarrow \ldots$

$\in [0, dim)$.