```
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;
}
```

!ok
≡ ok == 0
!ok == 1

ok == 0

i++;

Il nome delle variabili deve avere un significato (non per il C) per il lettore.

Quando scrivete programmi di grande dimensione il nome delle variabili deve dare una immediata conoscenza del significato della variabile.

NumeroDiValori
NumeroDiValoriPositivi
SommaDeiValori

```
int pippo(int gambadilegno,
          int minni[],
          int topolino)
{ int pluto = 0;
  int clarabelle = 0;
  while (pluto < topolino && !clarabelle)
    if (minni[pluto] == gambadilegno)
      :
```

08:56

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

  return trovato;
}
```

Controllare se un array ha tutti i valori diversi tra loro (distinti)

Dichiarazione ↓ eseguite in sequenza

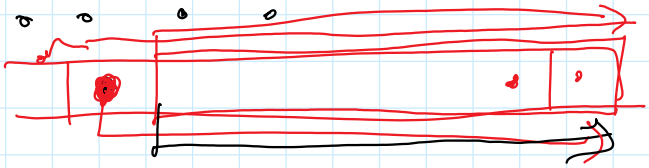
```
main ()
{

}
```

```
int member (...)
{
}
```

```
int distinct (...)
{
}
```

potrò usare la funzione member

Programmare in modo modulare



member più generale: cerca un elemento in un array, ma con indici di inizio e fine della ricerca parametrici

```
int member (int el, int a[], int inizio, int fine)
```

Cerca el in a nell'intervallo degli indici [inizio, fine)

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

calcola un esistenziale ∃

member (el, a, inizio, fine) ≡

$(\exists i \in [inizio, fine).\ el = a[i])$

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

```
if (!membro(a[i], a, i+1, dim)) i++;
else ok = 0;
```

```
{ ok = !membro(a[i], a, i+1, dim);
  i = i+1;
}
```

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

lunedì 1 ottobre 2018 10:21

Controllo che tutti gli elementi di un array siano diversi tra loro.

Come si può esprimere con una formula logica

$$distinti(a, dim) \equiv$$

$$(\forall i \in [0, dim-1).\ (\neg \exists j \in [i+1, dim).\ a[i] = a[j]))$$

lunedì 1 ottobre 2018 10:27

Dato un array scrivere una funzione che è vera se esiste un elemento minore del successivo. Funzione fino al penultimo. a $[0, dim)$

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

$$\equiv$$

$$minore(a, dim)$$

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

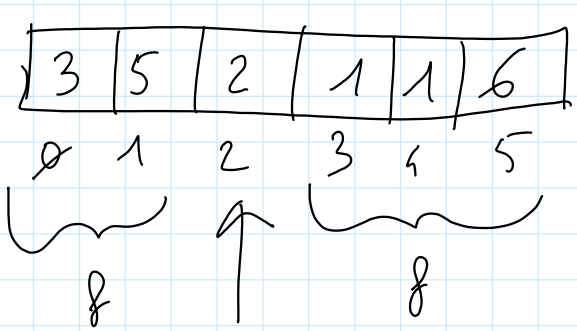
$$(\forall i \in [0, dim-1).\ a[i] < a[i+1])$$

$$\equiv \quad minori(a, dim)$$

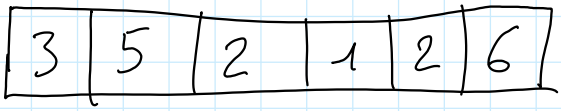
```
int minori (int a[], int dim)
{ int i = 0;
  int ok = 1;
  while (i < dim - 1 && ok)
    if (a[i] < a[i+1]) i++;
    else ok = 0;
  return ok;
}
```

a di dimensione dim

$$\left(\exists i \in [1, dim-1) . \sum_{j=0}^{i-1} a[j] = \sum_{j=i+1}^{dim-1} a[j]\right)$$



vale true - 1



vale false - ~~0~~

```
int somme (int a[], int inizio, int fine)
{ int i;
  int S = 0;
  for (i = inizio; i < fine; i++)
        S = S + a[i];
  return S;
}
int somma2 (int a[], int dim)
{ int i = 1;
  int trovato = 0;
  while (i < dim - 1 && !trovato)
     if (somme (a, 0, i) ==
         somme (a, i+1, dim)) trovato = 1;
     else i++;
  return trovato;
}
```

lunedì 1 ottobre 2018 10:57

Dati due array a e b, di numero di elementi dima e dimb, rispettivamente.

Esiste un elemento di a che compare anche in b?

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