giovedì 28 settembre 2017 16:03

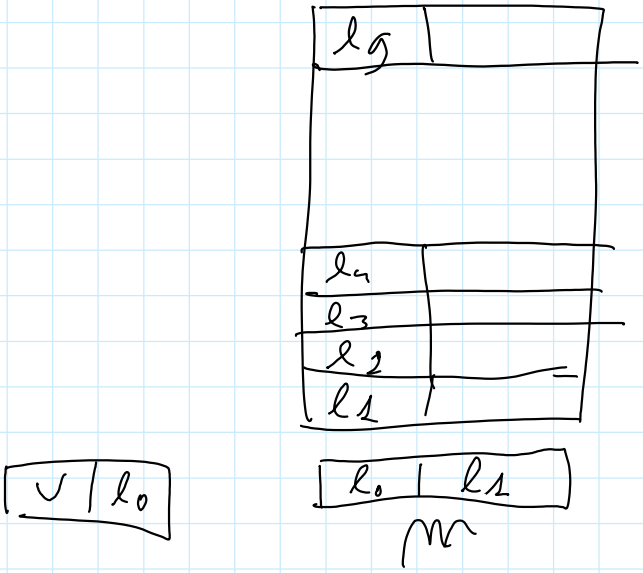
ARRAY – FUNZIONI E PROCEDURE

```
void read (int v[])
{
    for (int i=0; i<10; i++)
        scanf ("%d", &v[i]);
}
```

`&v[i]` → `v+i`

`v[i]` → `*(v+i)`

v ha come valore l'indirizzo del primo elemento dell'array.
v+i è l'indirizzo del primo elemento dell'array + i parole di memoria



$l_1 + i$ es: $i = 3$

$l_1 + 3 = l_4$

$l_1 + 0$ $\quad i = 0$

indirizzo del primo elemento

l_1	v

p	l_0

l_0	l_1

$x \longrightarrow l_1$

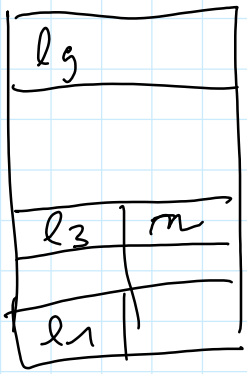
$*x \longrightarrow v$

$\mathcal{L}x \longrightarrow l_0$

↳ l'indirizzo del valore di x in memoria

$\mathcal{L}(*(v+i))$

$\mathcal{L}\,v[i]$



$\mathcal{L}(*(v+2)) = v+2$

v	l_0

l_0	l_1

$l_1 + 2 = l_3$

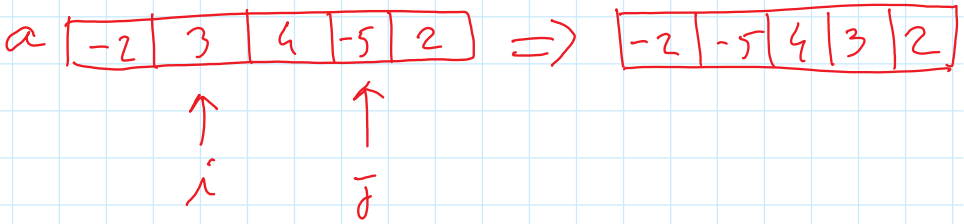
$*(l_3) = m$

$\mathcal{L}(m) = l_3$

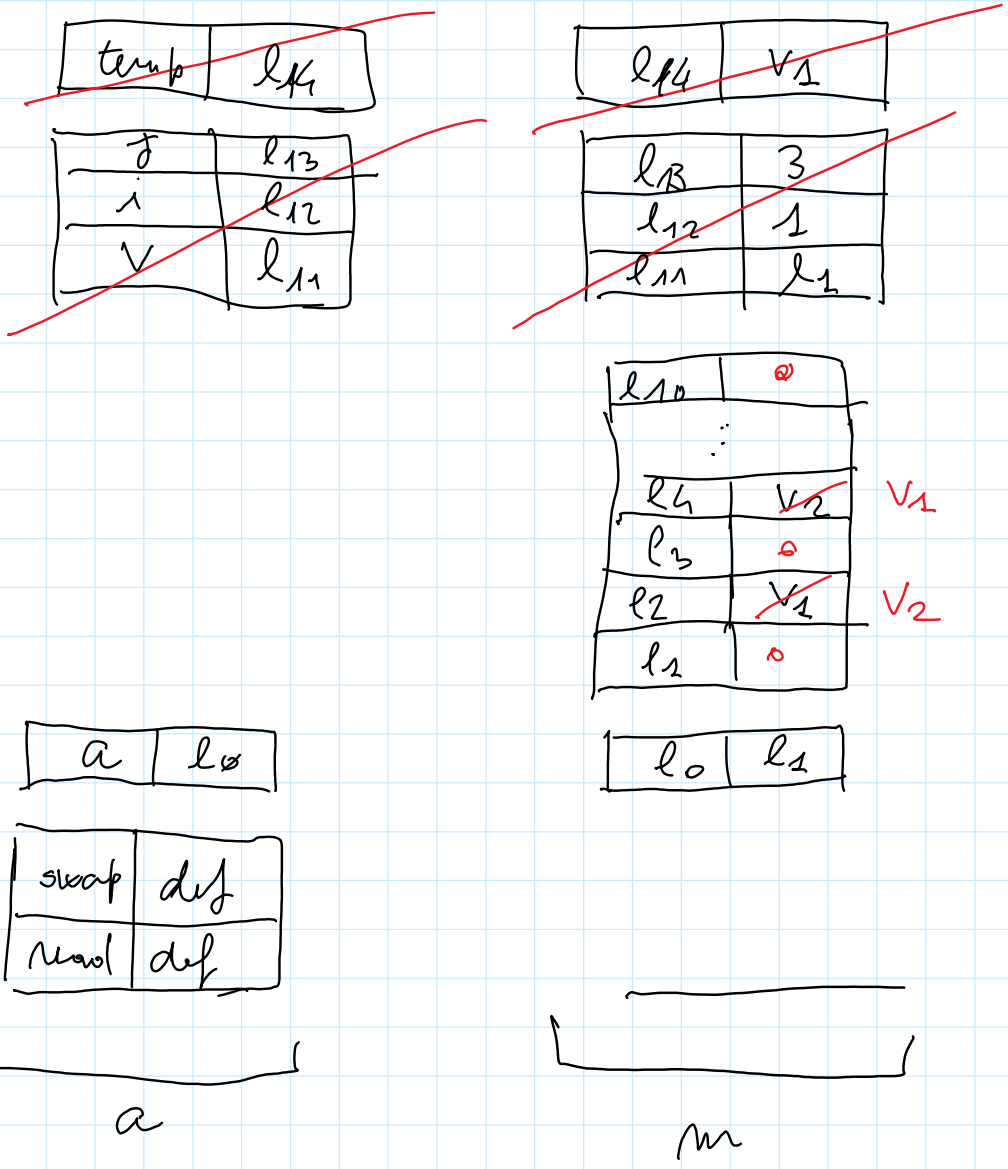
```
void swap(int v[], int i, int j)
{ int temp = v[i];
  v[i] = v[j];
  v[j] = temp;
}
```

FUNZIONA??

SI!!




```
main()
{ int a[10];
  read(a);
  swap(a, 1, 3)
```



$v[i]$

$*(v+i)$

$*(l_1+1)$

$*(l_2)$

$v[i] = v[j];$

$*(v+i) = *(v+j)$

$*l_2 = *l_4$

v_2

$v[j] = temp;$

$*(v+j) = temp$

$*(l_4) = temp$

$*(l_4) = temp$

giovedì 28 settembre 2017 16:03

```
#include <stdio.h>

void read (int v[])
{ for(int i=0; i<10; i++)
      scanf ("%d", &a[i]);
}

main()
{ int a[10];
  int b[15];
  read (a);
  read (b);
```

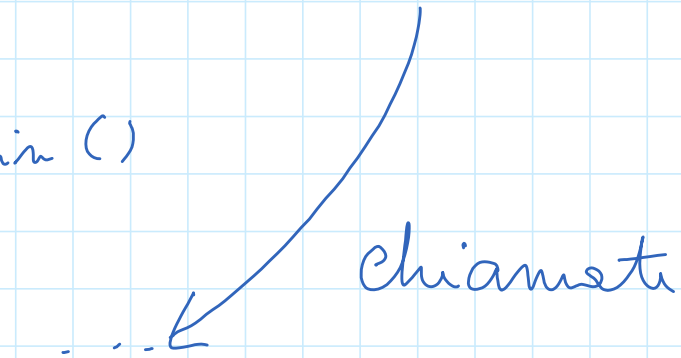
```
void read (int v[], int dim)
{ for (int i=0; i<dim ; i++)
      scanf ("%d", &v[i]);
}

main ()
{ int a[10];
  int b[15];
  read (a, 10);
  read (b, 15);
```

giovedì 28 settembre 2017 16:03

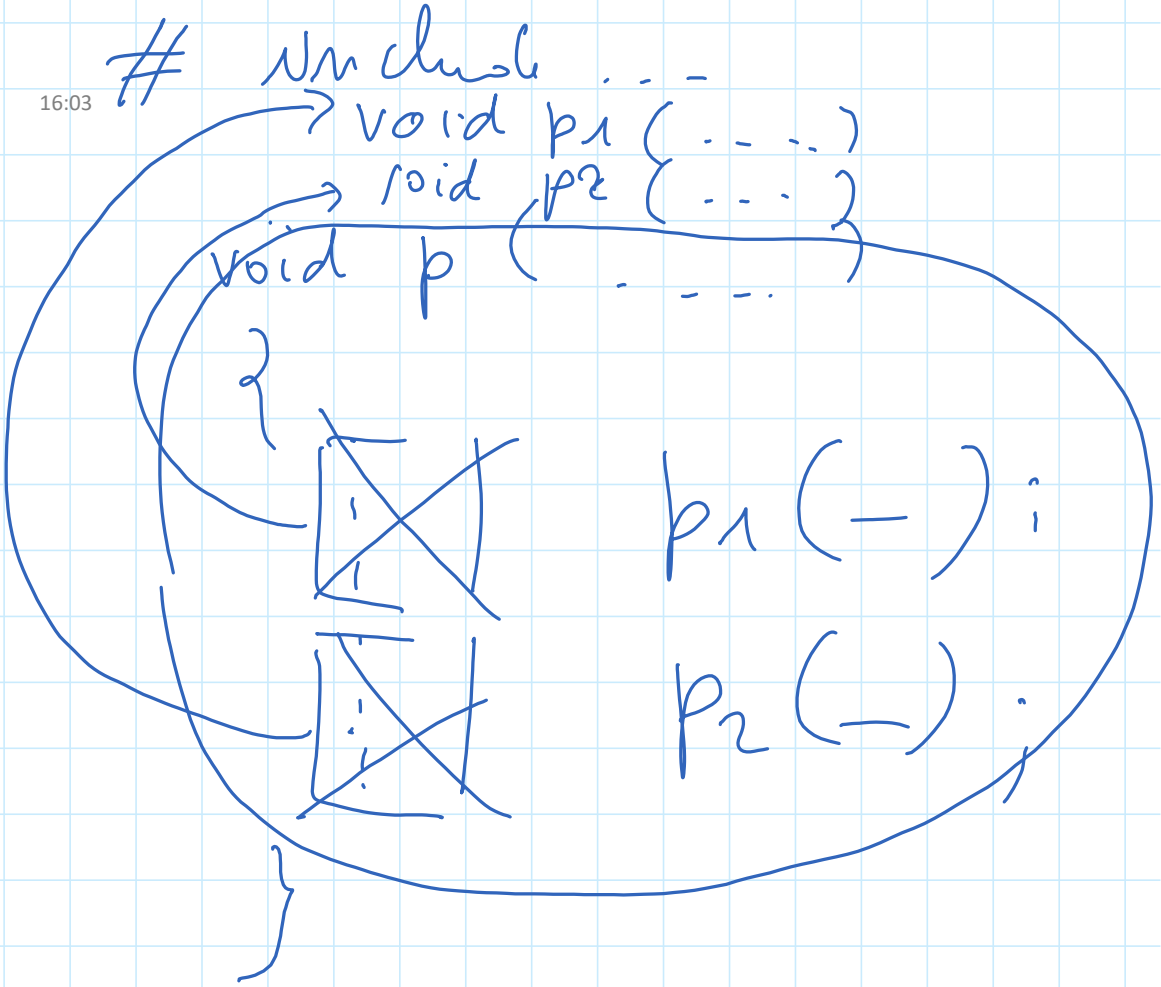
#include ...

{ Sequenze di dichiarazioni di funzioni o procedure

main()
{
...
}

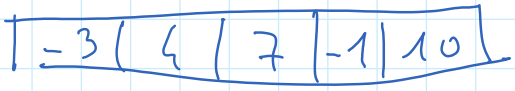


giovedì 28 settembre 2017 16:03

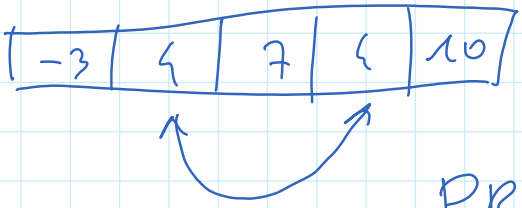


a 

Vedere se tutti gli elementi dell'array sono distinti (diversi tra loro)

a 

elementi distinti

a 

no!!

PROBLEM SOLVING

giovedì 28 settembre 2017 16:04

```
#include <stdio.h>
void read (int v[], int dim)
  { ... }
int member(int el, int v[], int inizio, int fine)
  { ... }
int distinti (int v[], int dim)
  { ... }
main()
{ int a[10];
  read (a, 10);
  if (distinti (a,10)) printf("tutti gli element sono distinti");
  else printf("non tutti gli element sono distinti");
}
```

Scriviamo una funzione

```
int member (int el, int v[], int inizio,
            int fine)
{ int i = inizio;
  int trovato = 0;
  while (i < fine && !trovato)
```

```
        if ( v[i] == el) trovato = 1;
          else  i++;
    return trovato;
}
```

```
int menuhen (int el, int v[], int inizio, int fine)
{ int trovato = 0;
  for (int i = inizio; i < fine; i++)
    if (a[i] == el) trovato = 1;
    else trovato = 0;
```

```
{ int trovato;
  for (int i = inizio; i < fine; i++)
    if (a[i] == el) trovato = 1;
```

- Ok, ma inefficiente

giovedì 28 settembre 2017 16:04

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

distinti == 1

Scrivere una funzione che restituisca la somma di tutti i valori maggiori di 0 in un array.

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

void sum(int v[], int dim, int *s)
{ *s = 0;
  for(int i=0; i<dim; i++)
    if(v[i]>0) *s = *s + v[i];
}
```

giovedì 28 settembre 2017 16:04

da lunedì siamo in AULA E Telediattica