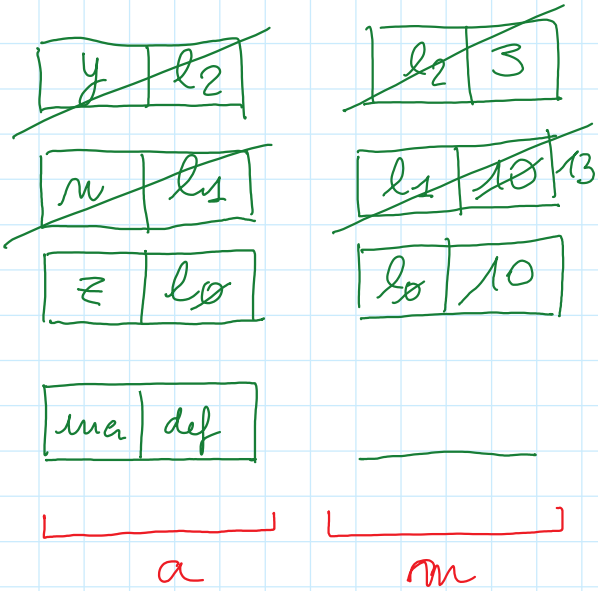
```
void inc (int n)
{ int y = 3;
  n = n + y;
}

main ()
{ int z = 10;
  inc (z);
}
```

13

comando

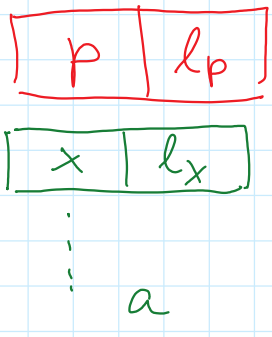
PROCEDURE

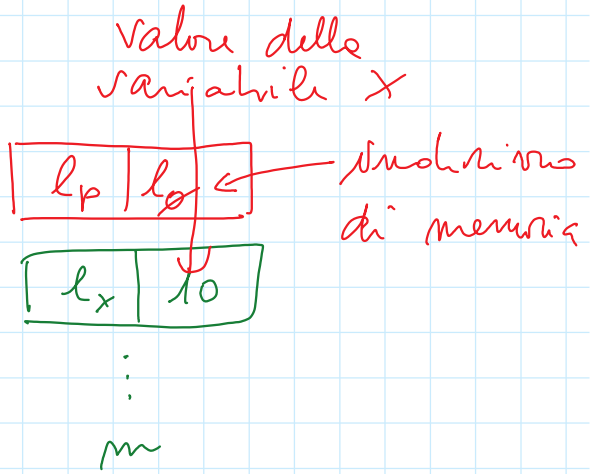


PUNTATORI

Variabile puntatore è una variabile il cui valore è un indirizzo di memoria.

```
int x = 10;
```





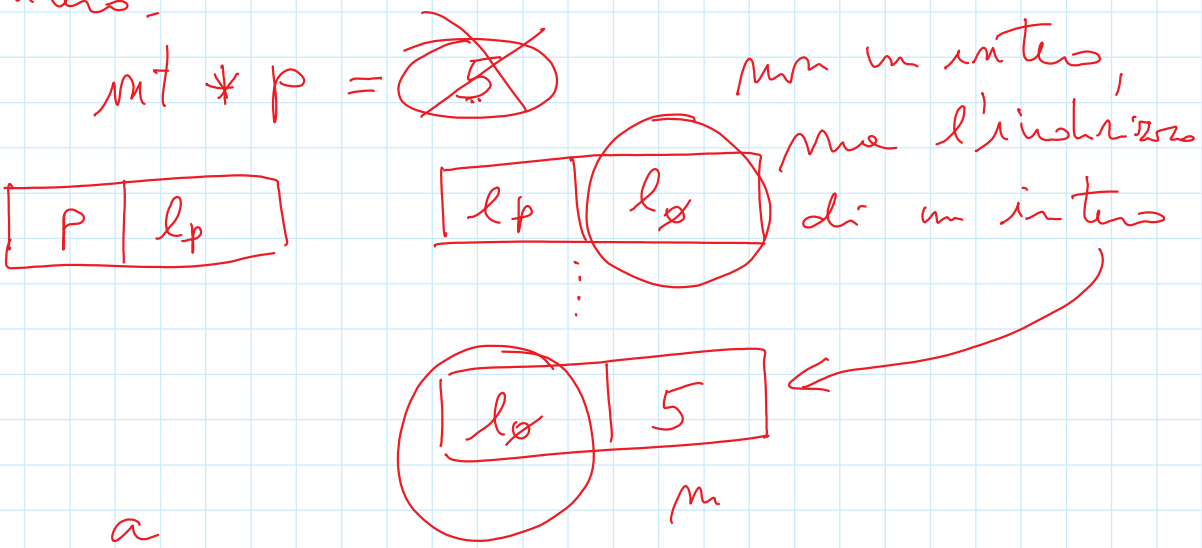

```
int * p
```

la variabile p che sto dichiarando non avrà in memoria un valore INTERO ma l'indirizzo di memoria che conterrà un intero.

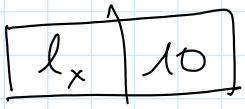
es:

```
int * p = 5;
```

non un intero, ma l'indirizzo di un intero



&x non il valore di x, ma l'indirizzo di memoria associato a x nell'ambiente



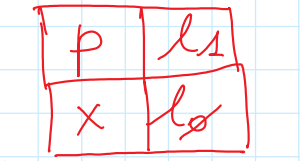
x ha valore 10
&x ha valore l_x

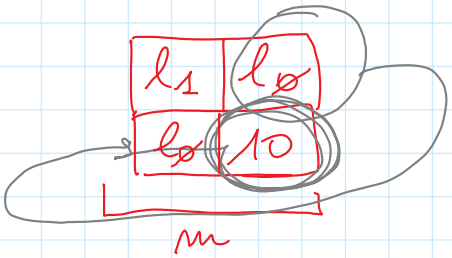
a

m

main()

```
{ int x = 10;
  int* p = &x;
  ...
  *p = *p + 1;
```



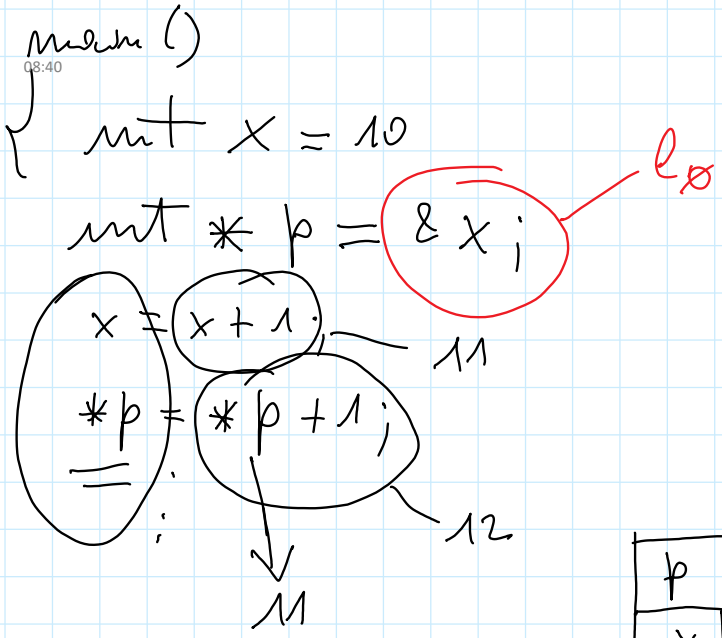


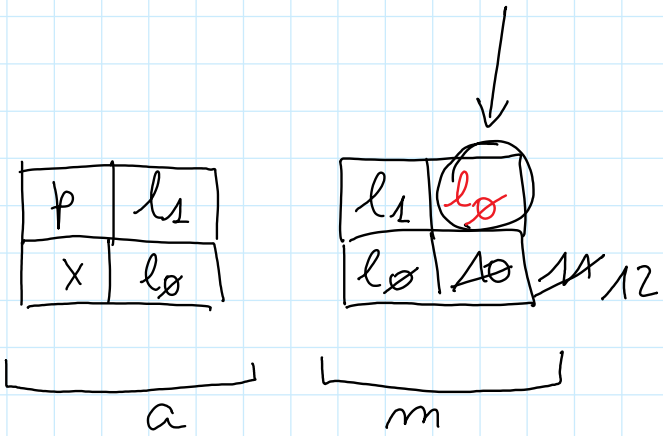
10

non voglio il valore di p ($l_\emptyset$) ma voglio il valore puntato (il valore che sta in memoria all'indirizzo che è valore di p) che è p.

Non assegnare il valore 10+1 a p, ma alla locazione di memoria che è valore di p ($l_\emptyset$)

$(l_\varnothing)$


main ()
{
int x = 10
int * p = &x ;
l_0
x = x + 1 ;
11
*p = *p + 1 ;
12
;
11

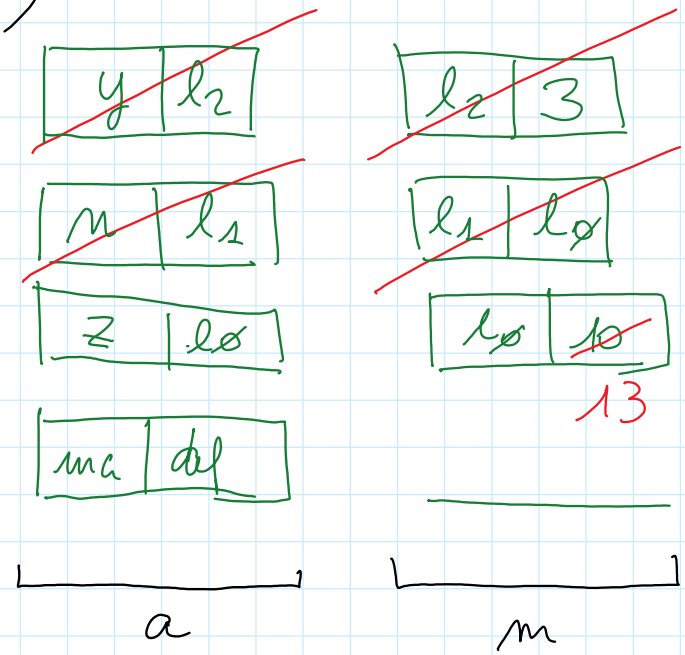

p
l_1
x
l_0
l_1
l_0
l_0
10
11
12
a
m

```
void mia (int * n)
{ int y = 3;
  *n = *n + y;
}

main()
{ int z = 10;
  mia (&z);
}
```

13

10

3



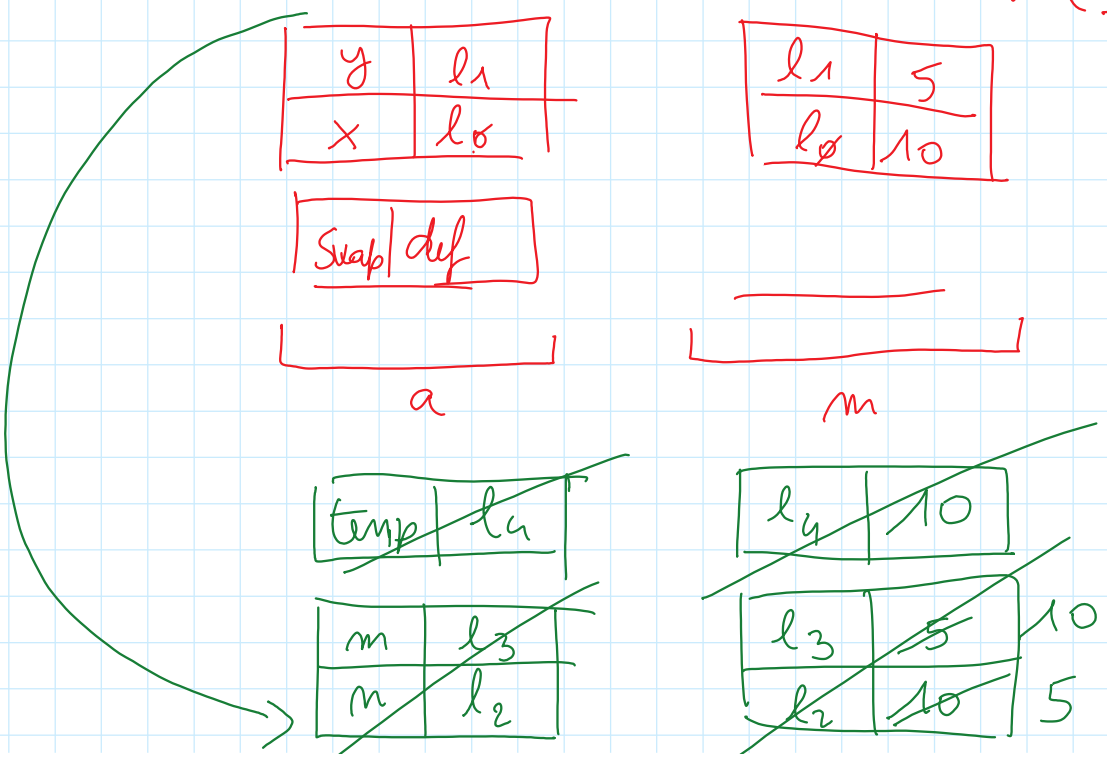
```
main()
{ int x = 5;
  int y = 10;
  x = y;
  y = x;
```

```
main()
{ int x = 5;
  int y = 10;
  int temp = x;
  x = y;
  y = temp;
  :
```

ok

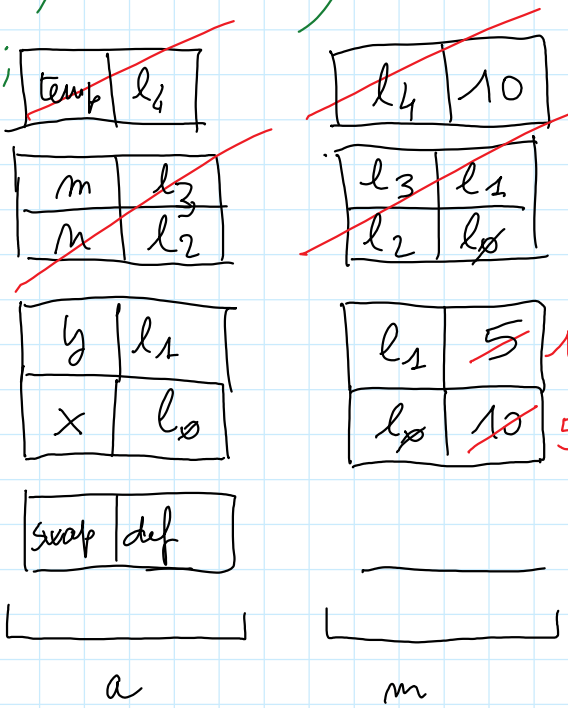
```
void swap(int n, int m)
{ int temp = n;
  n = m;
  m = temp;
}
```

```
main()
{ int x = 10;
  int y = 5;
  swap(x, y);
```




n
l_2
l_2
10
5

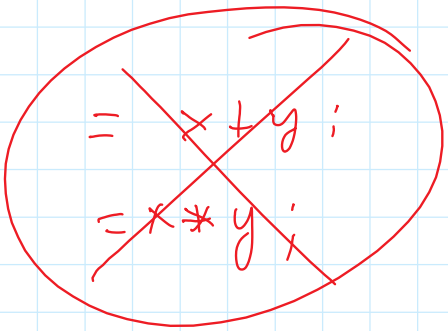
```
void swap (int *n, int *m)
{ int temp = *n;
  *n = *m;
  *m = temp;
}
main ()
{ int x = 10;
  int y = 5;
  swap (&x, &y);
```



lunedì 25 settembre 2017 08:40

Date due variabili dobbiamo scrivere una funzione o procedura che calcola la somma e il prodotto delle due variabili

```
main()
{   int x = 10;
    int y = 5;
    int somma;
    int prodotto;
```

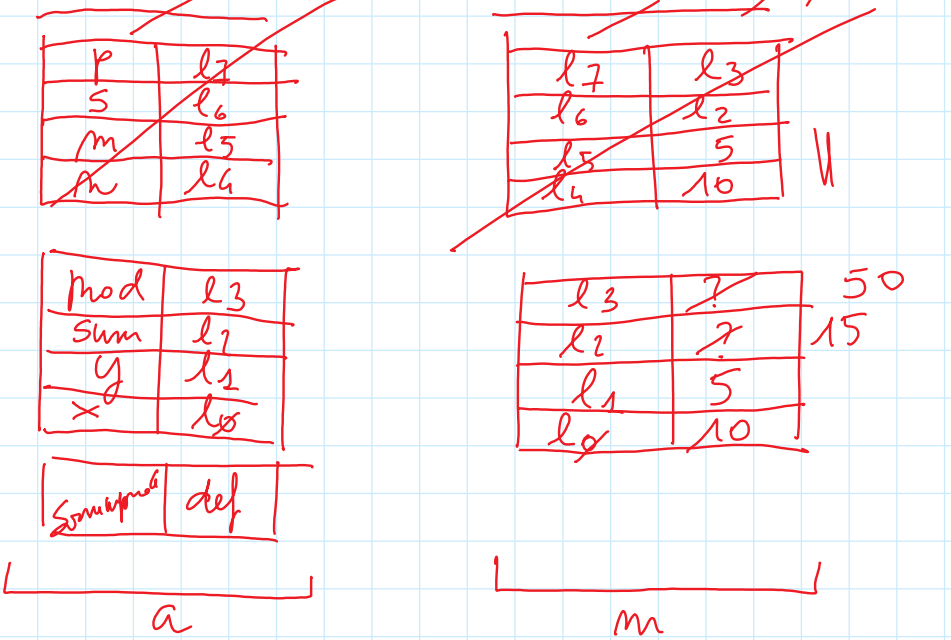


lunedì 25 settembre 2017 08:40

due

```
int somma (int n, int m)
{ return n + m;
}

int prodotto (int n, int m)
{ return n * m;
}

main ()
{ int x = 10;
  int y = 5;
  int s = somma (x, y);
  int p = prodotto (x, y);
```

```
void sommaprod (int n, int m, int *s, int *p)
{ *s = n+m; *p = n*m; }

main()
{ int x = 10;
  int y = 5;
  int sum;
  int prod;
  sommaprod (x, y, &sum, &prod);
```



lunedì 25 settembre 2017 08:40

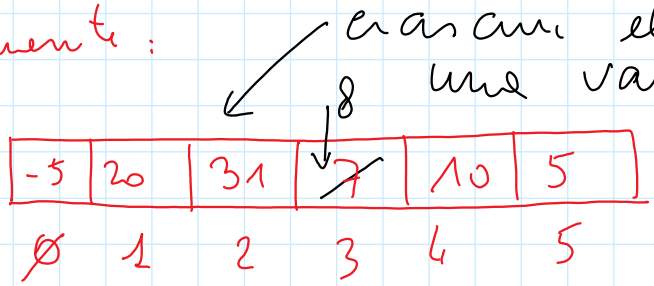
```
int sommaprod (int n, int m, int *p)
{
    *p = n * m;
    return n+m;
}

main()
{
    int x = 10;
    int y = 5;
    int prod;
    int sum;
    sum = sommaprod(x, y, &prod);
}
```

ARRAY

lunedì 25 settembre 2017 08:40

Un array è una struttura dati di elementi dello stesso tipo, ciascuno individuato da un indice.

Graficamente:



interi

Gli indici vanno da 0 al numero degli elementi meno 1.

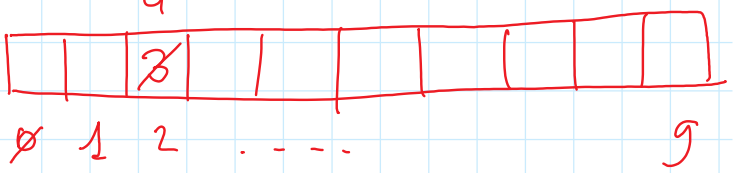
Dichiarazione di array:

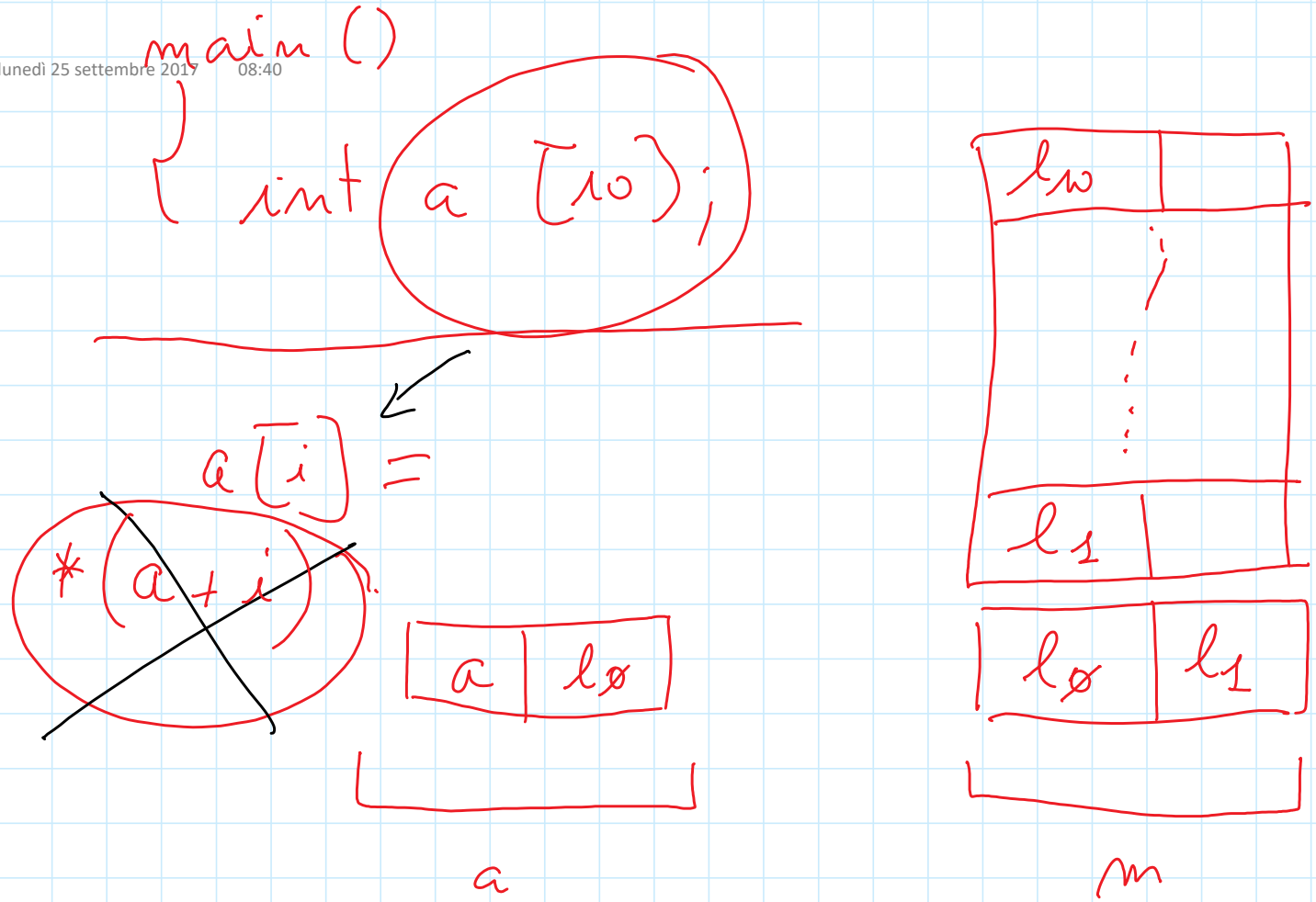
```
int a [10]
```

tipo degli elementi — nome — numero degli elementi (deve essere una costante)

```
main()
{ int a[10];
  a[2] = 3;
  a[2] = a[2] + 1;
}
```

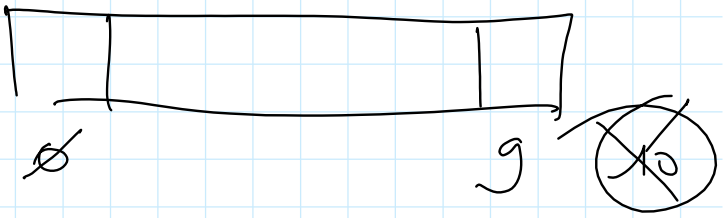
4




main ()
{ int a [10];
a [i] =
* (a + i)
a
l0
a
l0
l1
m

lunedì 25 settembre 2017 08:40

```
main ()
{ int a [10];
  a[0] = 0;
  a[1] = 0;
   .
   .
   .
  a[9] = 0;
```

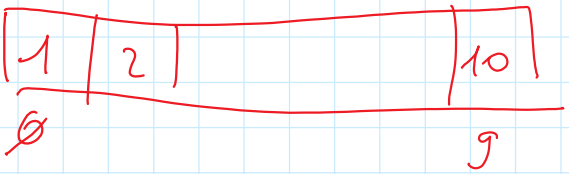


a[10] non esiste

lunedì 25 settembre 2017 09:51

```
main()
{ int a[10];
  for(int i=0; i<10; i++)
  { a[i] = 0; }
}
```

$i = i + 1$

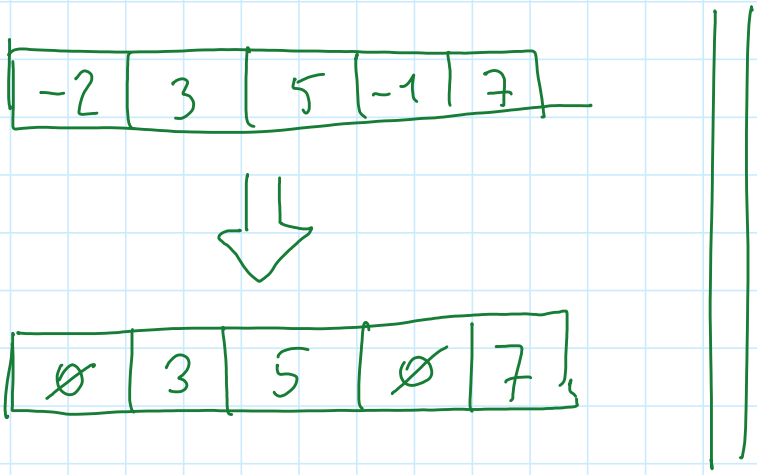



```
a[i] = i+1;
```

lunedì 25 settembre 2017 08:40

Dato un array, vogliamo azzerare tutti gli elementi < 0

es.




```
main()
{ int a[10];
```

Lettura dell'array $i<10$ / $=$

```
for(int i=0; i<=9; i++)
    if(a[i]<0) a[i]=0;
```

⋮

Cercare un elemento in un array

lunedì 25 settembre 2017 08:40

```
main()
{ int a[10];
  int el = 5;      <-- l'elemento da cercare
  int i = 0;
  Lettura dell'array
  while (i < 10 && a[i] != el)
      i = i + 1;
```

- $i = 10$ non l'ho trovato
- $i < 10$ l'ho trovato

l'and, che in C si indica con && (in logica $\wedge$), è un operatore che si applica a due valori logici (true e false) e vale true sse entrambi gli argomenti sono true

```
int fact (int n)
{ int f=1;
  while (n>1) {
      f = f*n;
      n = n-1;
  }
  return f;
}
```

PROCEDURE → void

non c'è n--;

FUNZIONE

```
main()
{ int x=10;
  x = fact(x);
}
```