

```
int * x;
int y;
```

il valore delle variabile non è un intero ma è l'indirizzo di memoria di un valore intero

```
x = y;
```

errore

Ambiente e Memoria servono per **ESPLICITARE** gli indirizzi di memoria (locazioni)

y	l <sub>1</sub>
x	l <sub>0</sub>

Ambiente

l <sub>1</sub>	5
l <sub>0</sub>	l <sub>1</sub>

Memoria

Non può essere un valore intero

l<sub>1</sub> è l'indirizzo di memoria di un valore intero

(in questo caso 5)

In C c'è la possibilità di indicare, oltre al valore, l'indirizzo di una variabile.

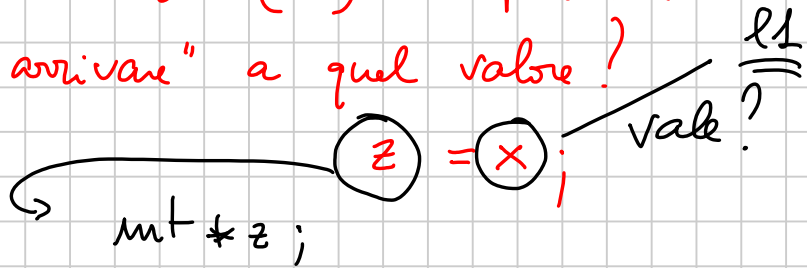
```
int * x;
int y = 5;
```

$x = \&y;$

l'indirizzo delle variabile y (cioè le locazioni associate a y nell'ambiente)

CORRETTO

Se x ha come valore l'indirizzo di un intero (5) è possibile "arrivare" a quel valore?



y	l1
x	l2

Amb.

l1	5
l2	?

Mem

l1

Come "arrivare" da una variabile puntatore al valore "puntato"?

Mediante l'espressione

$*x$

il valore intero puntato da  $x$ ,  
cioè il valore intero che ha come  
indirizzo il valore di  $x$

Qual'è il valore di:

$x \rightarrow l_1$

$*x \rightarrow 5$

$\&x \rightarrow l_0$

```
int *x;  
int y = 5;  
x = &y;
```

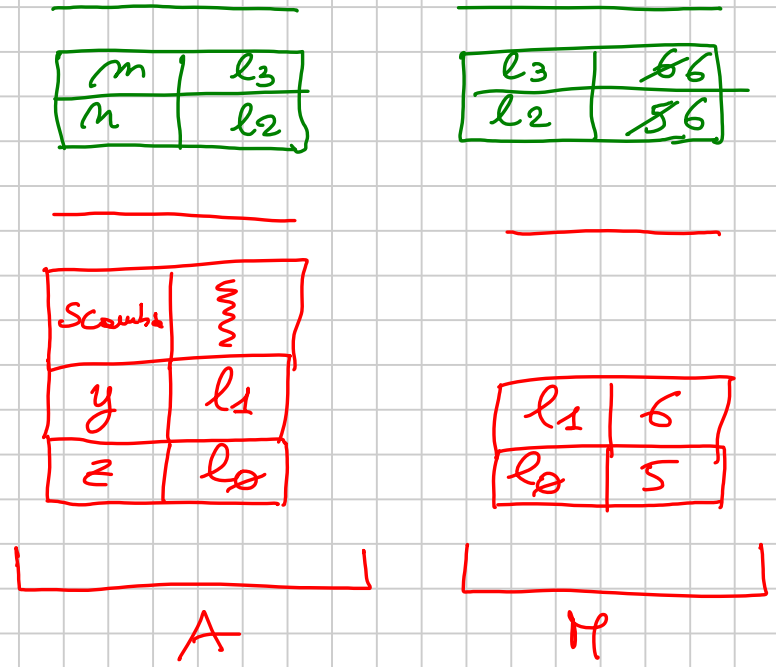
y	$l_1$
x	$l_0$

$l_1$	5
$l_0$	$l_1$

```

int z = 5; ←
int y = 6; ←
void scambia (int m, int n) //
{ ←
  m = m; // NO
  n = n; //
}
→
main ()
{ ←
  scambia (z, y); ←
  //
}

```



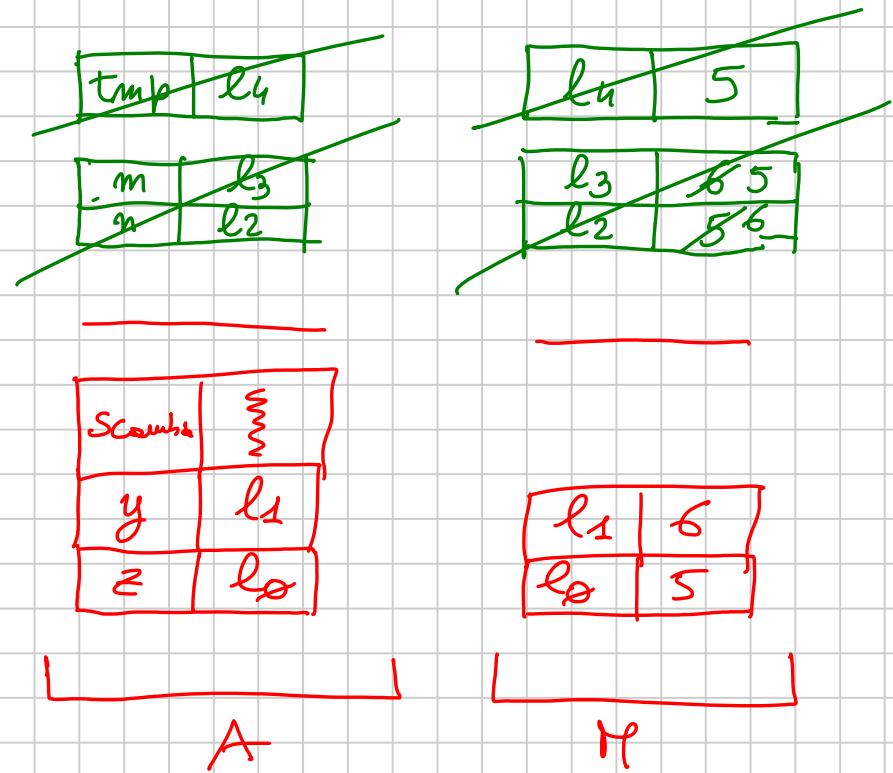
```

int z = 5; ←
int y = 6; ←
void scambia (int n, int m)
{
    int tmp = n;
    n = m;
    m = tmp;
}
main ()
{
    scambia (z, y); ←
}
    
```

Diagram illustrating the function call in main():

- Function call: `scambia(z, y);`
- Argument `z` (value 5) is passed to parameter `n`.
- Argument `y` (value 6) is passed to parameter `m`.
- The values 5 and 6 are circled in green.

PASSAGGIO DEI PARAMETRI IN C  
è "PER VALORE"



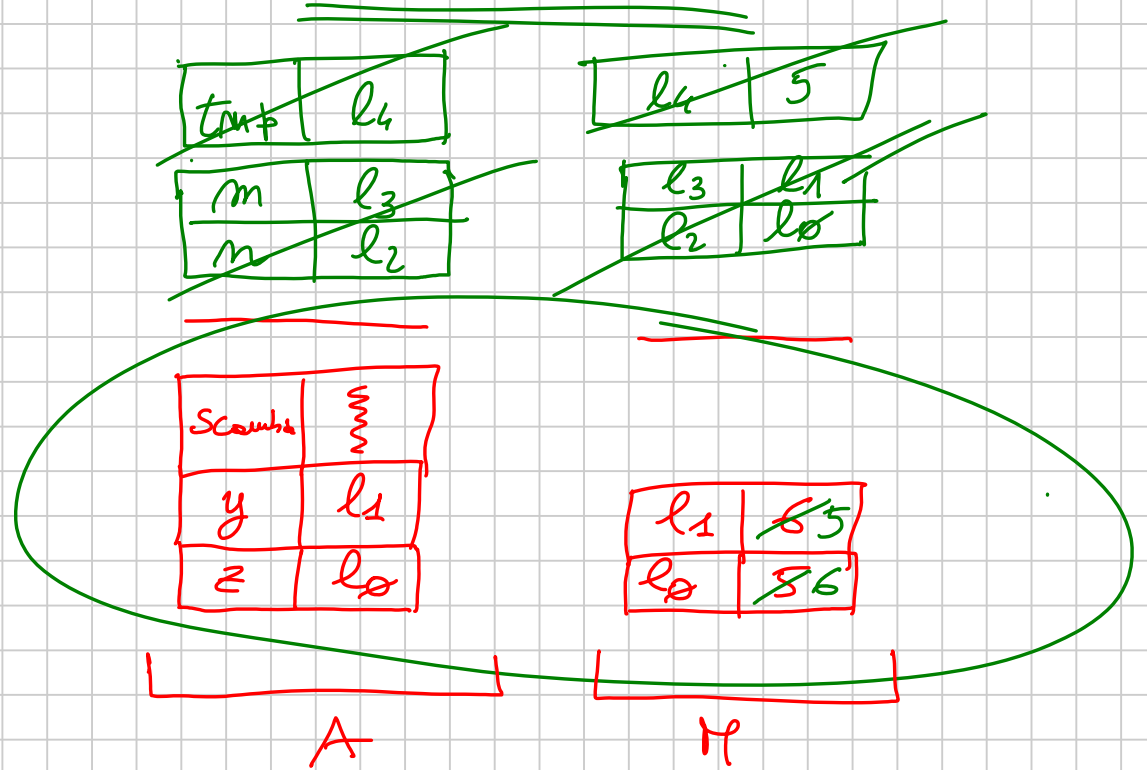
```

int z = 5;
int y = 6;
void scambia (int *m, int *n)
{
  int tmp = *m;
  *m = *n;
  *n = tmp;
}
main ()
{
  scambia (&z, &y);
}

```

Diagram illustrating the execution of the swap function. In the function call, `&z` and `&y` are passed to `*m` and `*n` respectively. Inside the function, `*m` points to `z` (address `l0`) and `*n` points to `y` (address `l1`). The value 5 is stored in `z` and 6 in `y`. The function uses a temporary variable `tmp` to swap the values: `tmp = *m` (5), `*m = *n` (6), and `*n = tmp` (5). The final state shows `z` containing 6 and `y` containing 5.

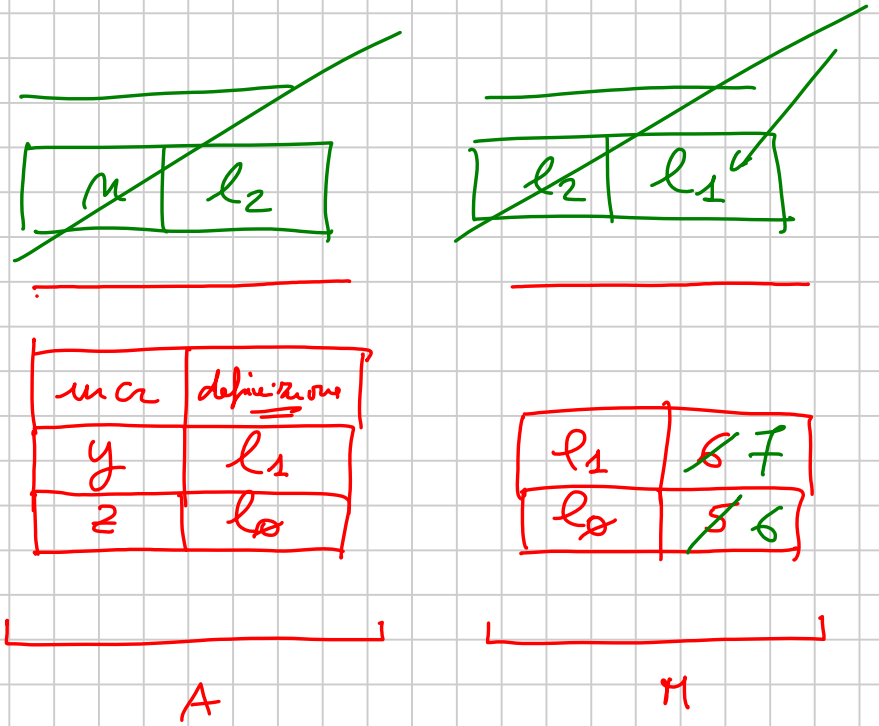
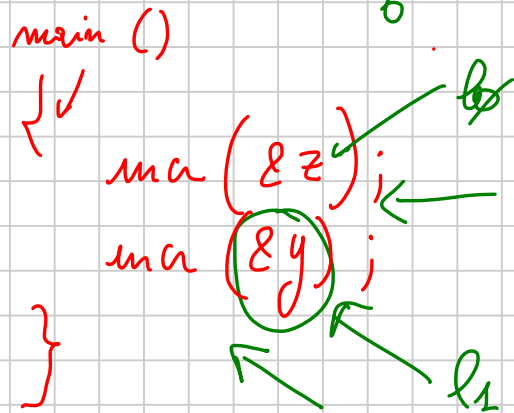
## PASSAGGIO DEI PARAMETRI IN C "PER VALORE"



```

int z = 5;
int y = 6;

void mca (int * m)
{
    * m = * m + 1;
}
    
```



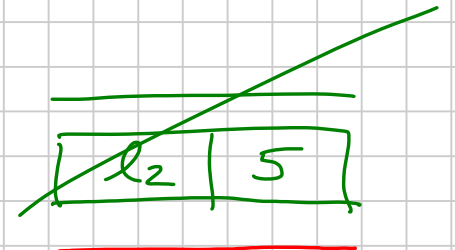
```

int z = 5;
int y = 6;
int mca (int m)
{ return m+1; }
    
```

```

main ()
{
    z = mca (z);
    y = mca (y);
}
    
```

Diagram: A green circle around 'z' in the first call is connected by a line to the number '6'. A green arrow points from the 'y' parameter in the second call to the 'return m+1' line in the function definition.



mca	definzioni
y	l1
z	l0

l1	87
l0	86

A

M



```

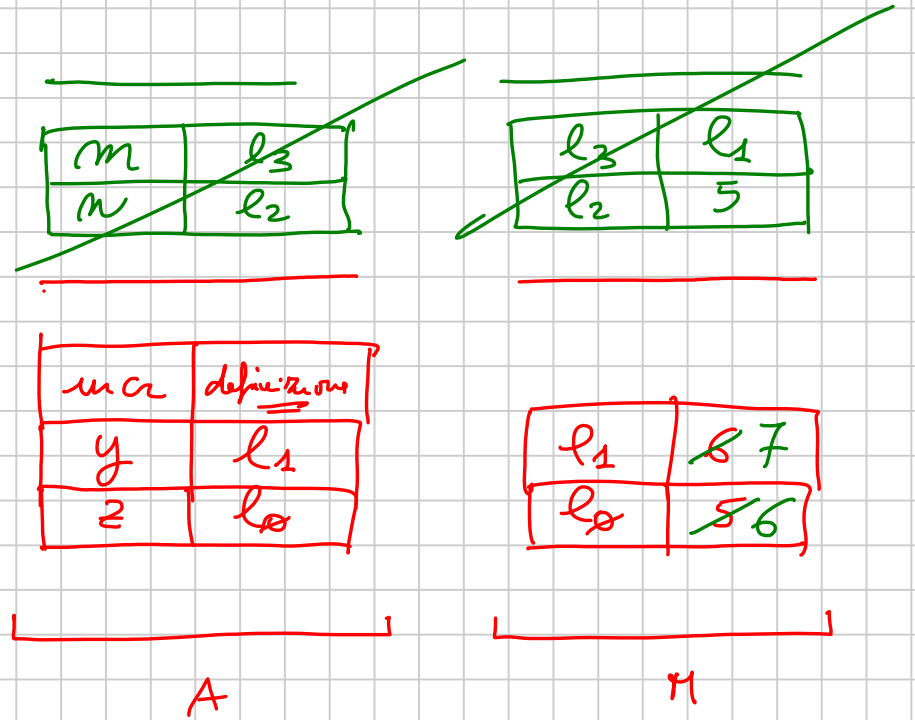
int z = 5;
int y = 6;
    
```

```

int incr (int m, int *m)
{
    *m = *m + 1;
    return m + 1;
}
    
```

```

main ()
{
    z = incr (z, &y);
}
    
```



```

int z = 5;
int y = 6;
    
```

```

int incr (int m int *m)
{
    *m = *m + 1;
    return m + 1;
}
    
```

```

main ()
{
    y = incr(y, &y);
}
    
```

m	l3
m	l2

l3	l1
l2	6

incr	definitions
y	l1
z	l0

l1	<del>6</del> 7
l0	5

A

M

```

int z = 5;
int y = 6;
    
```

```

int incz (int m, int *m) {
    *m = *m + 1;
    return y + 1;
}
    
```

```

main () {
    y = incz(y, &y);
}
    
```

Diagram showing memory addresses:  $l_1$  points to  $y$  (value 6),  $l_2$  points to  $z$  (value 5), and  $l_3$  points to the return value of `incz` (value 8).

### variabile "globale"

non usare variabili globali  
(non sono leggibili)

m	l <sub>3</sub>
n	l <sub>2</sub>

l <sub>3</sub>	l <sub>1</sub>
l <sub>2</sub>	6

incz	definizione
y	l <sub>1</sub>
z	l <sub>2</sub>

l <sub>1</sub>	<del>6</del> 7 8
l <sub>2</sub>	5

A

M

```

int x = 5;
int z = 6;
int y = 7;
int sum (int m)
{ return x + y + m; }

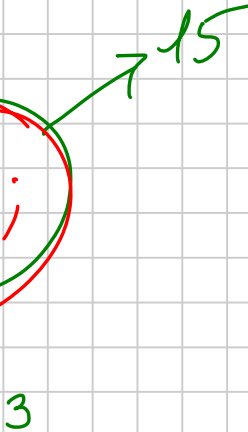
```

main ()

```

z = sum (3);

```



sum	m
z	l2
y	l1
x	l0

A

l2	6	15
l1	7	
l0	5	

A

# PORTATA DEGLI IDENTIFICATORI (SCOPE) STATICA

```

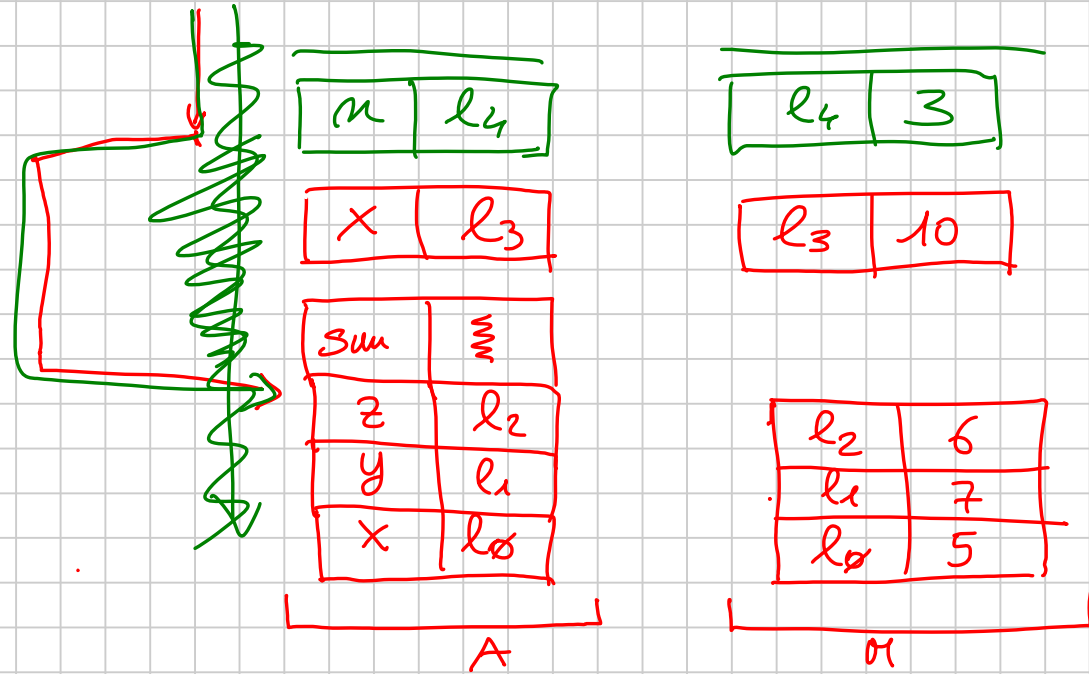
{
  int x = 5;
  int z = 6;
  int y = 7;
  int sum (int m)
  {
    return x + y + m;
  }
}
    
```

```

main ()
{
  int x = 10;
  z = sum (3);
}
    
```

~~x vale 10~~

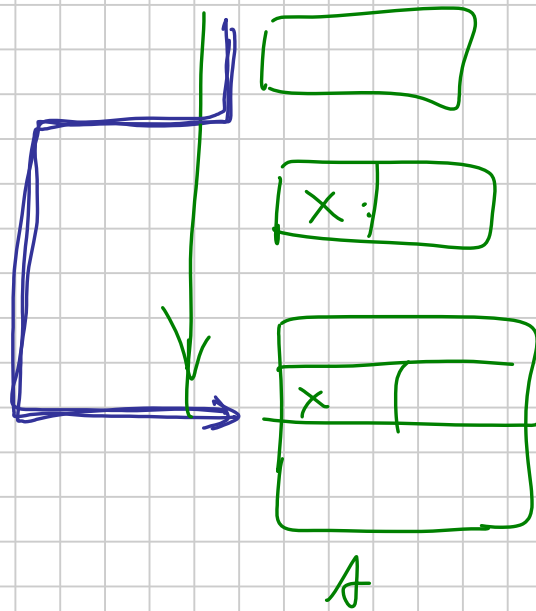
Le variabili globali vanno cercate nell'ambiente al momento della dichiarazione della funzione



Linguaggi in cui le variabili globali si cercano  
 scendendo le file nel modo usuale (dall'alto  
 verso il basso)

PORTATA DEGLI IDENTIFICATORI DINAMICA

costante  
 statica



# PORTATA DEGLI IDENTIFICATORI (scoping) STATICA

24/09/2015

```
{  
  int x = 5;  
  int z = 6;  
  int y = 7;  
  int sum (int m)  
  {  
    return m + 2;  
  }  
}
```

```
main ()  
{  
  int x = 10; 12  
  x = sum(x);  
}
```

