

Function & procedure C

---

array

---

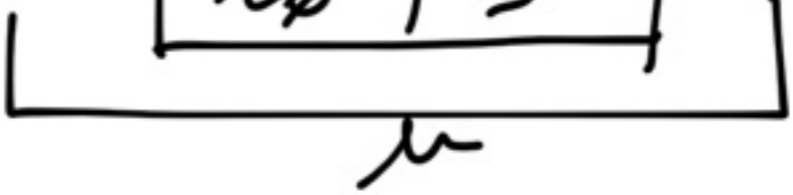
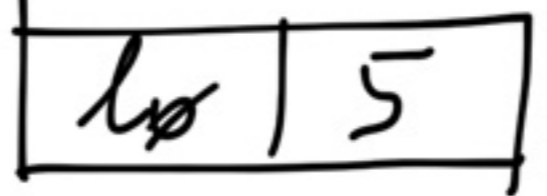
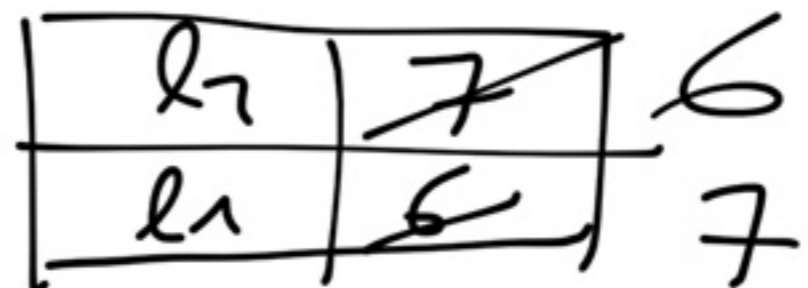
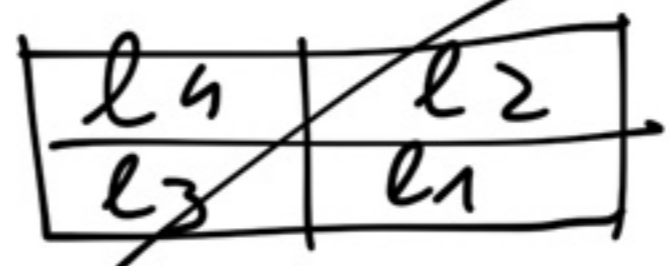
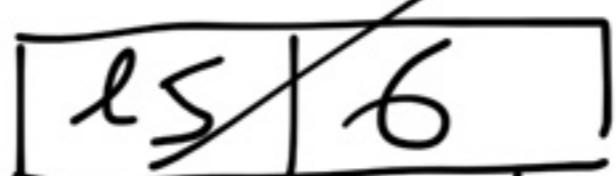
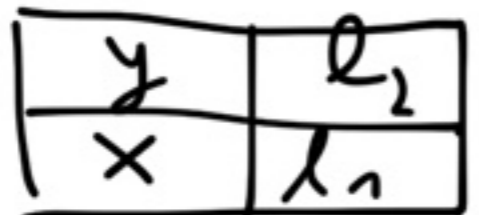
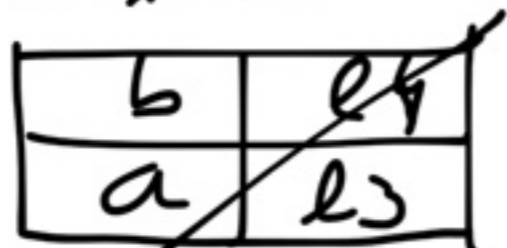
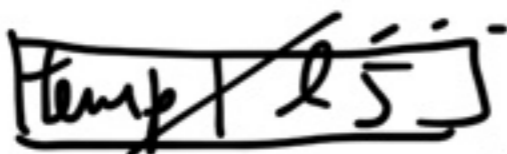
```

void swap (int * a , int * b)
{
  int temp = *a; *a = *b; *b = temp;
}

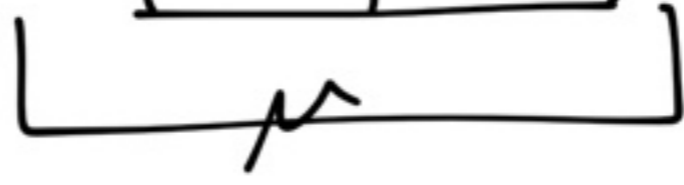
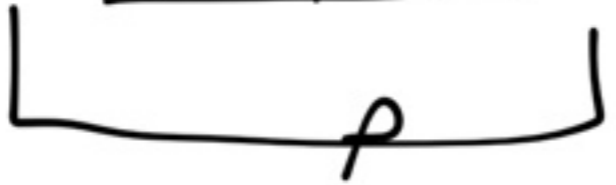
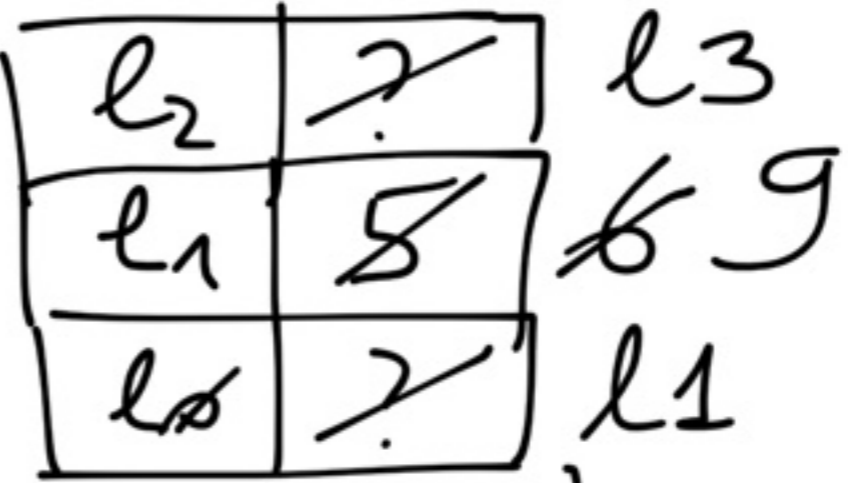
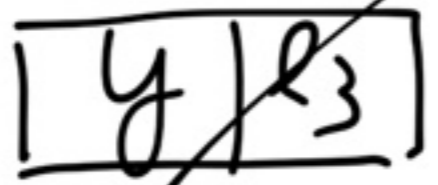
```

int x = 5;

int x = 6; int y = 7; swap (&x, &y);



$\{ \text{int } *p; \text{ int } x = 5; \text{ int } *p1;$   
 $\{ \text{int } y = 5; \text{ int } p1 = \&y;$   
 $x = x + 3; \text{ valori di } p \text{ e } p1?$   
 $*p1 = : *p1 \dots$

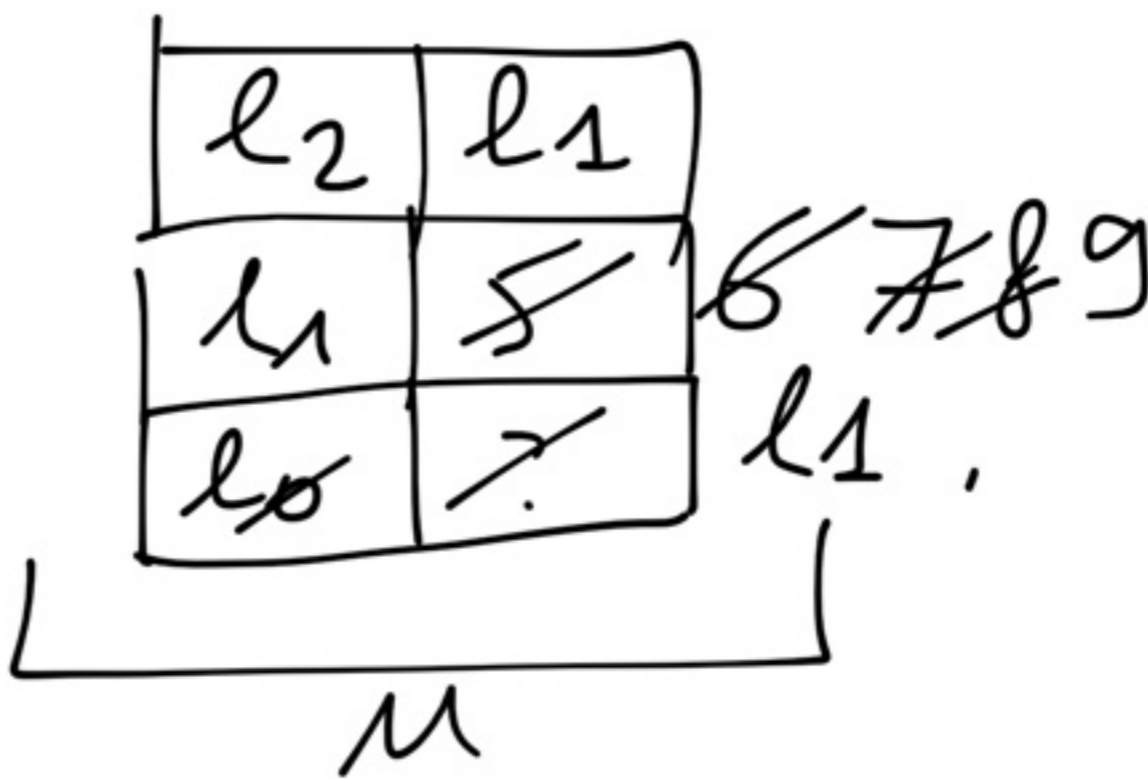
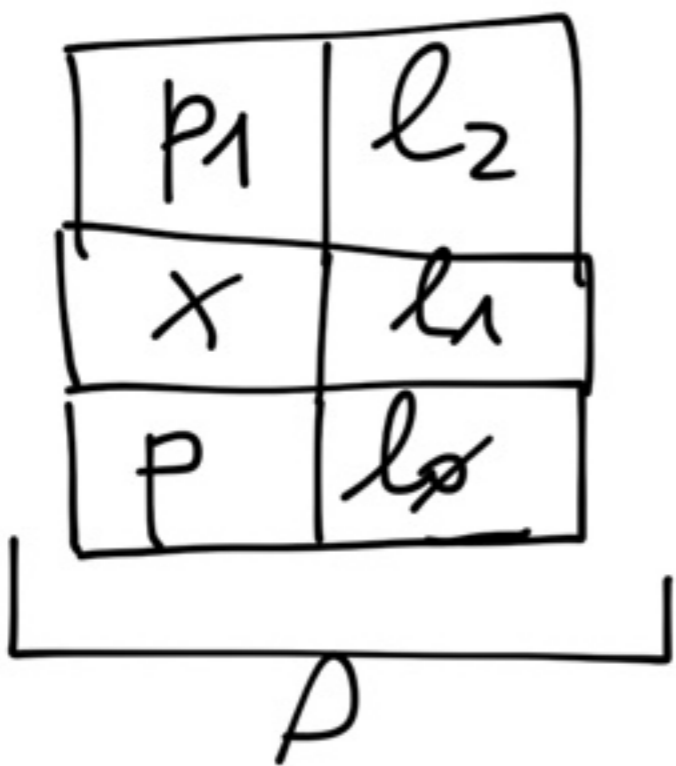


# DANGLING REFERENCES

---

$$\left. \begin{aligned}
 & m + *p; \quad m + x = 5; \\
 & m + *p1 = 8x; \\
 & p = p1; \quad *p = *p + 1; \\
 & *p1 = *p1 + 1; \quad *p = *p1 + 1;
 \end{aligned} \right\}$$

$$\underbrace{x = x + 1; \quad \dots}_{\dots} \quad \underbrace{*p = *p1 + 1; \quad \dots}_{\dots}$$



```

int x = 5;
void p(int a) { x = a + x; }

```

```

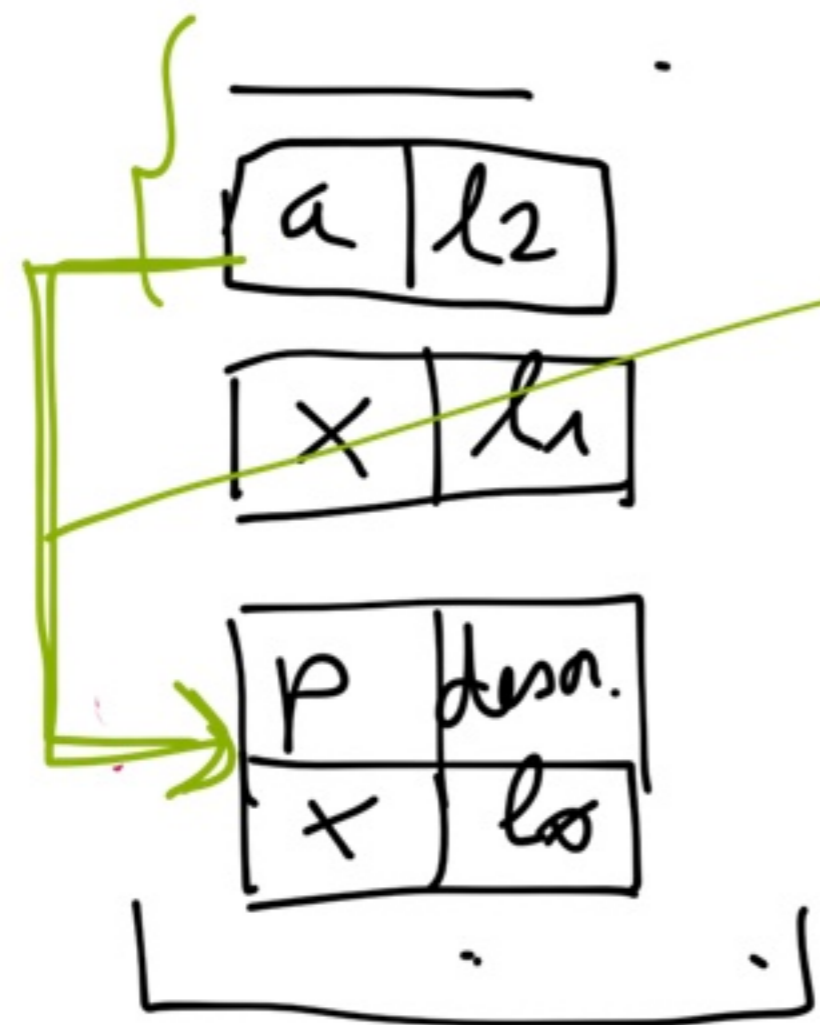
p(5);
{ int x = 15; p(x);
  ...
}

```

30

25

CATENA STATICA



30

~~10~~ 25

Regole per le variabili globali (regole di protezione degli identificatori)

- le variabili globali di una procedura sono quelle dell'ambiente di dichiarazione (statiche)

- le variabili globali di una procedura sono quelle dell'ambiente di chiamata (dinamiche)

le migliori parti dei  
ling. imperiali ma  
le regole STATICA



```

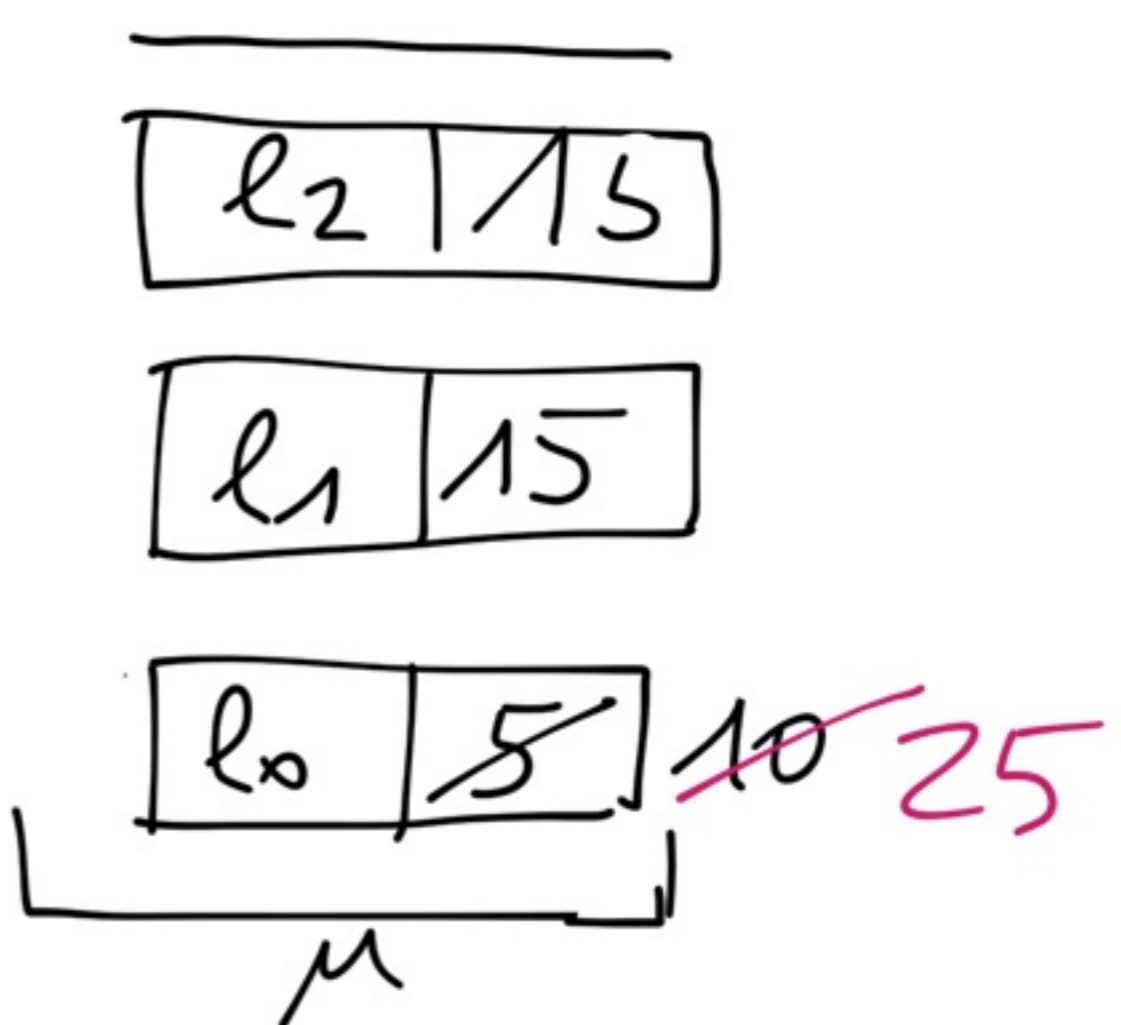
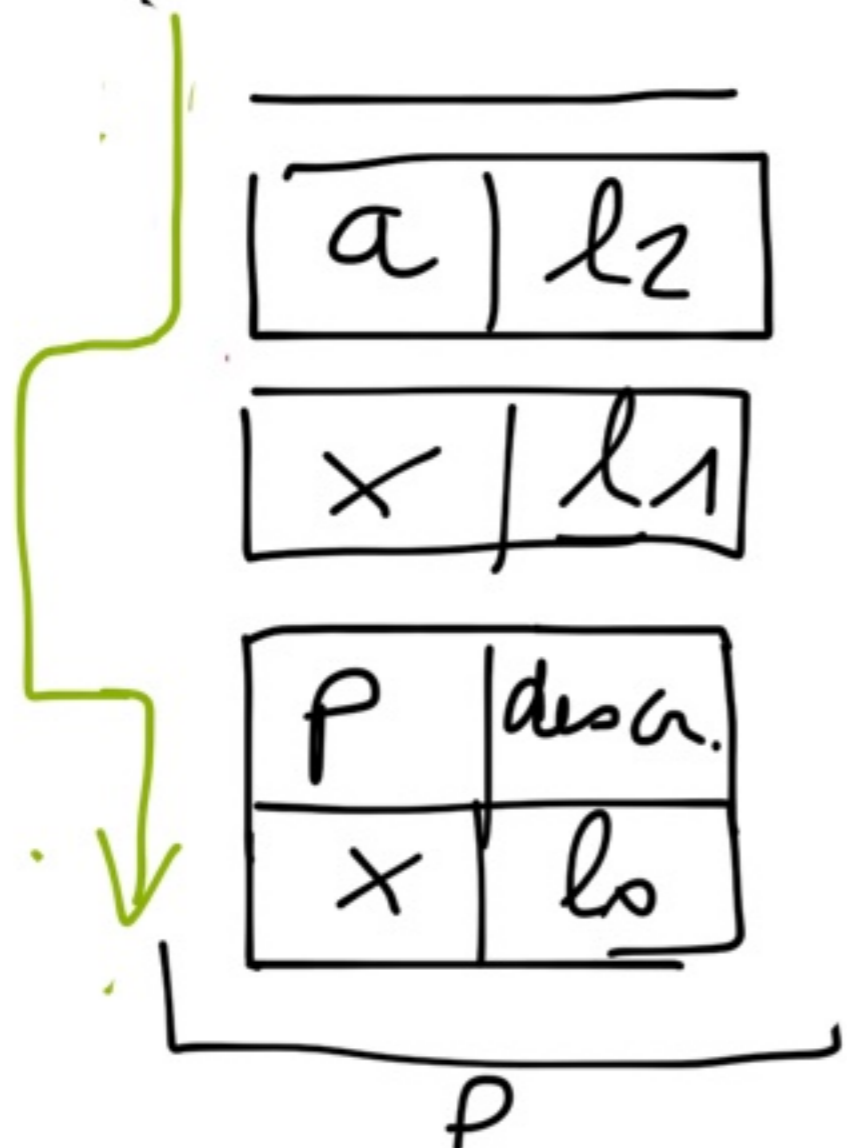
int x = 5,
void p(int a) { x = a + x; }
p(5);

```

```

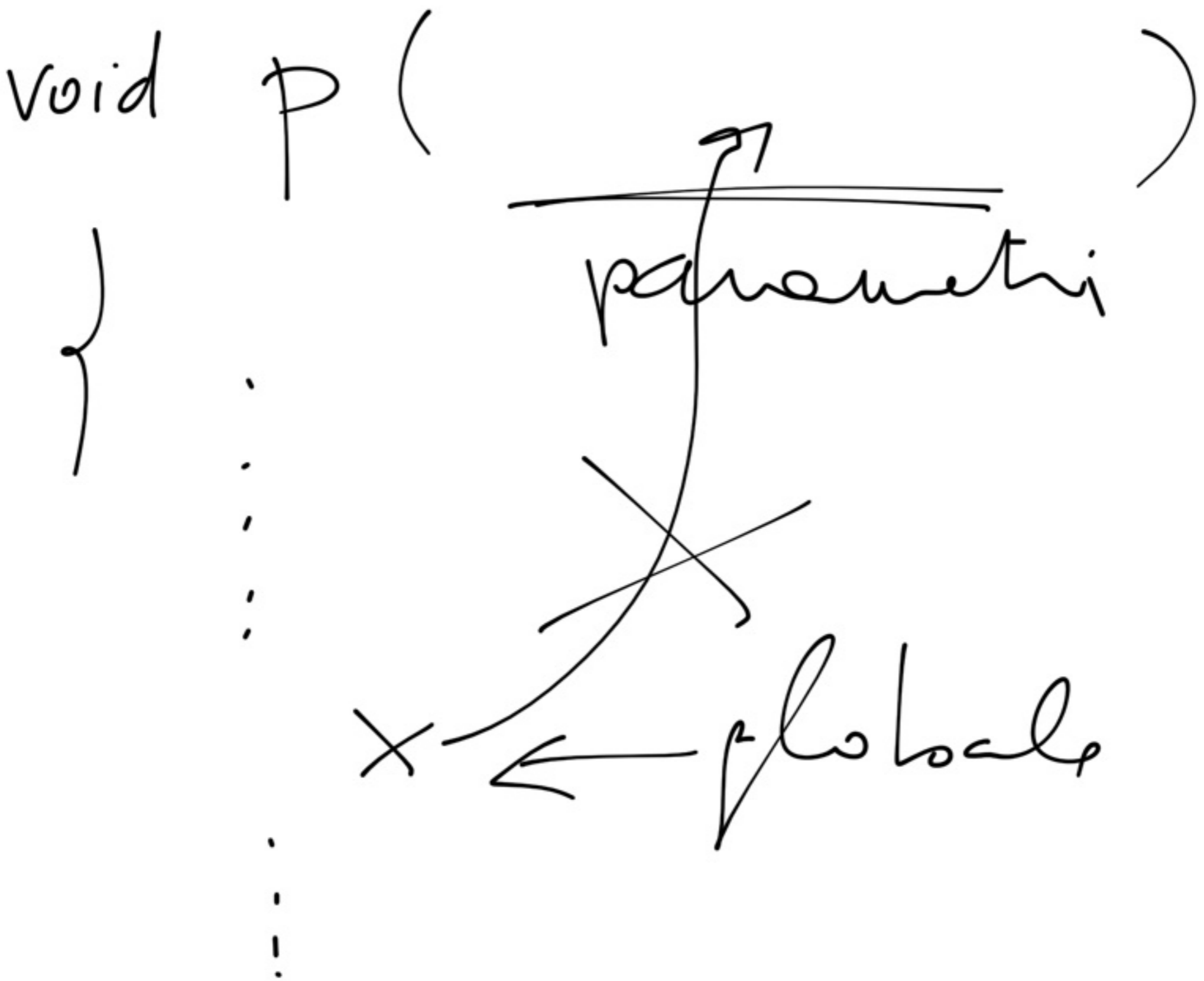
int x = 15; p(x),
...

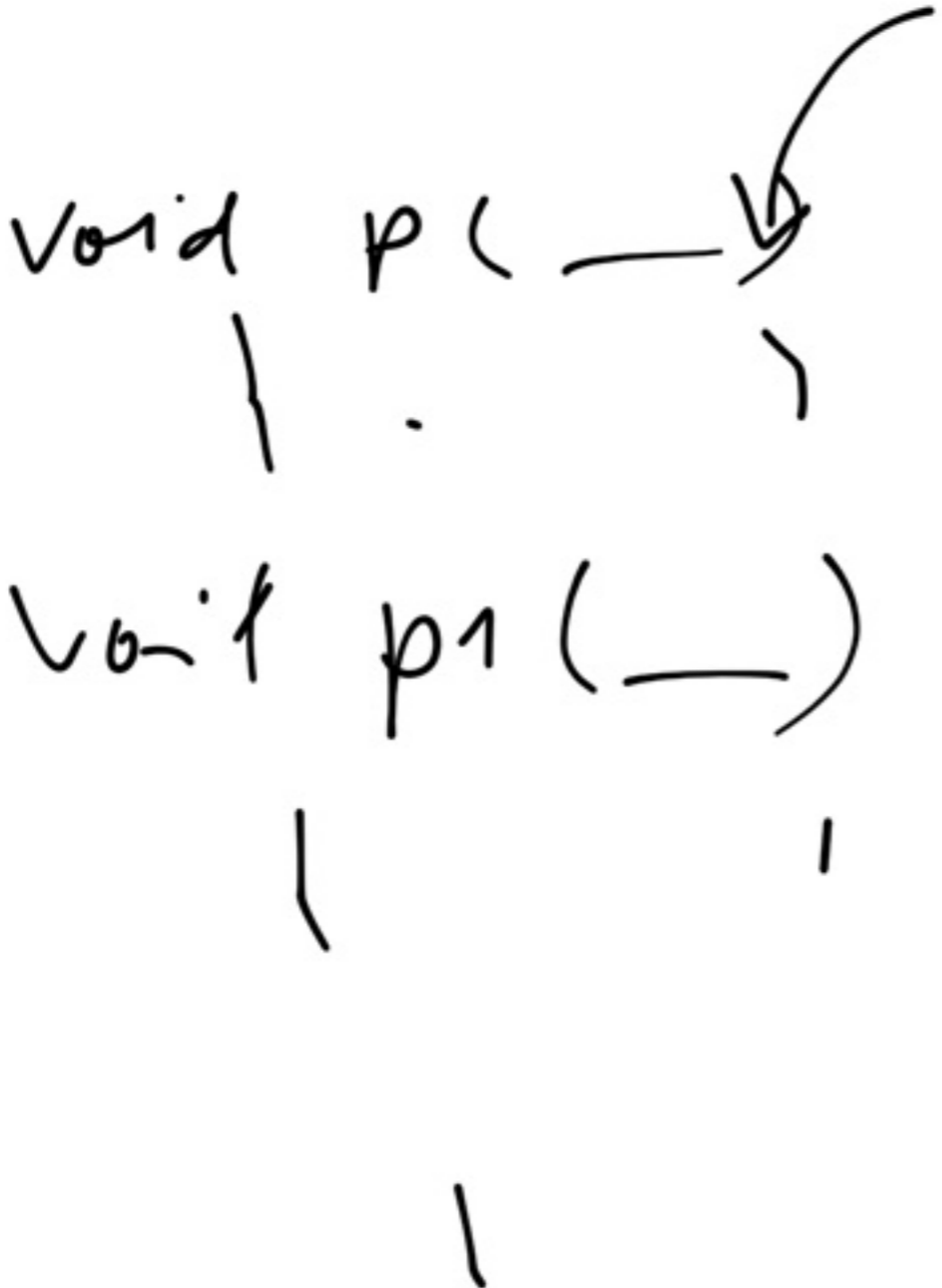
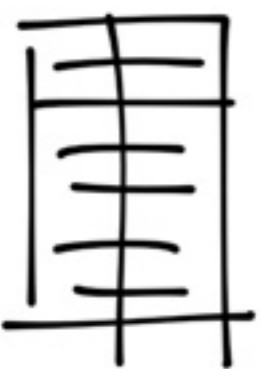
```



Regole d'uso

quante usanze  
sanitarie globali?





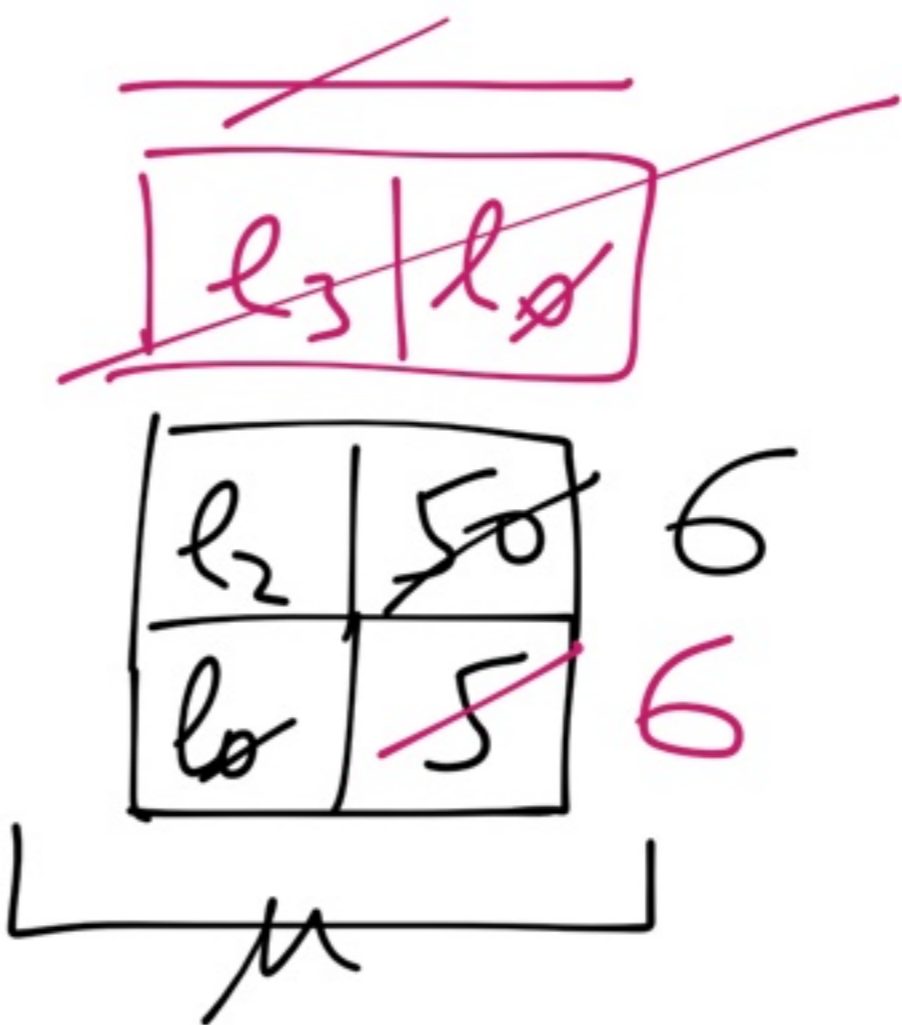
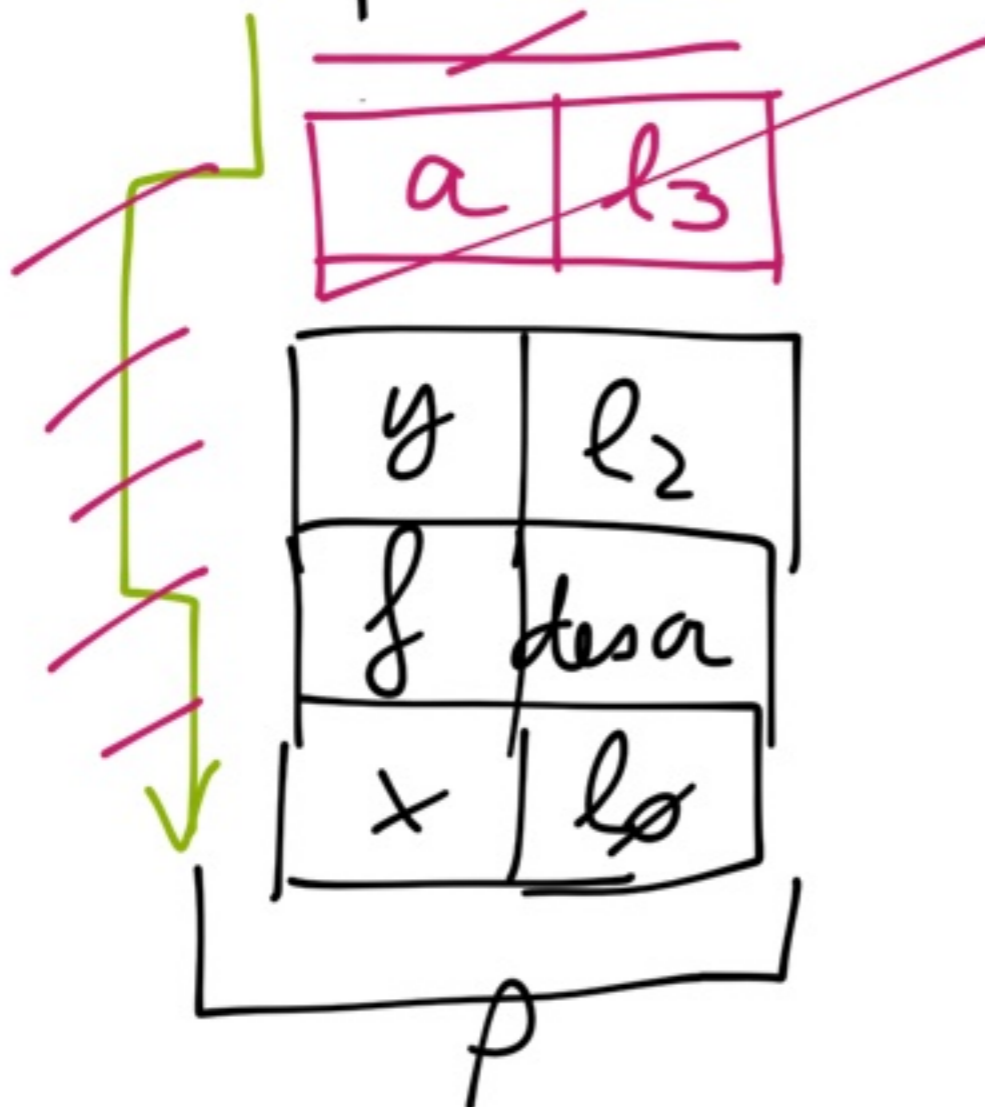
```
int f ( int * a )  
{  
    *a = *a + 1;  
    return *a;  
}
```

```

int x = 5;
int f(int *a) { *a = *a + 1; return *a; }
int y = 50;

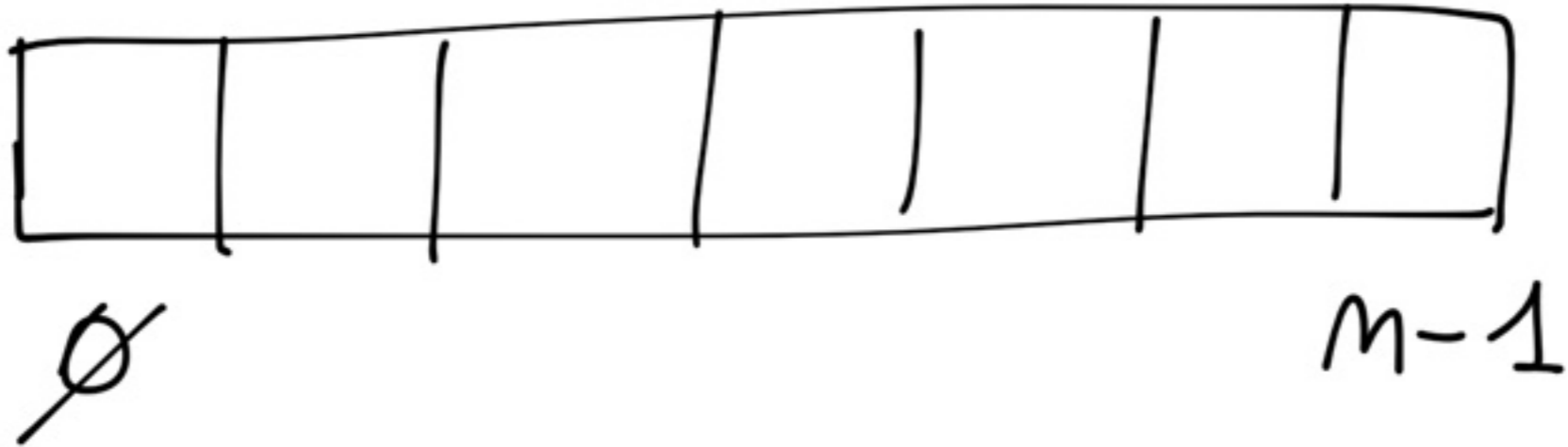
```

$y = f(x)$

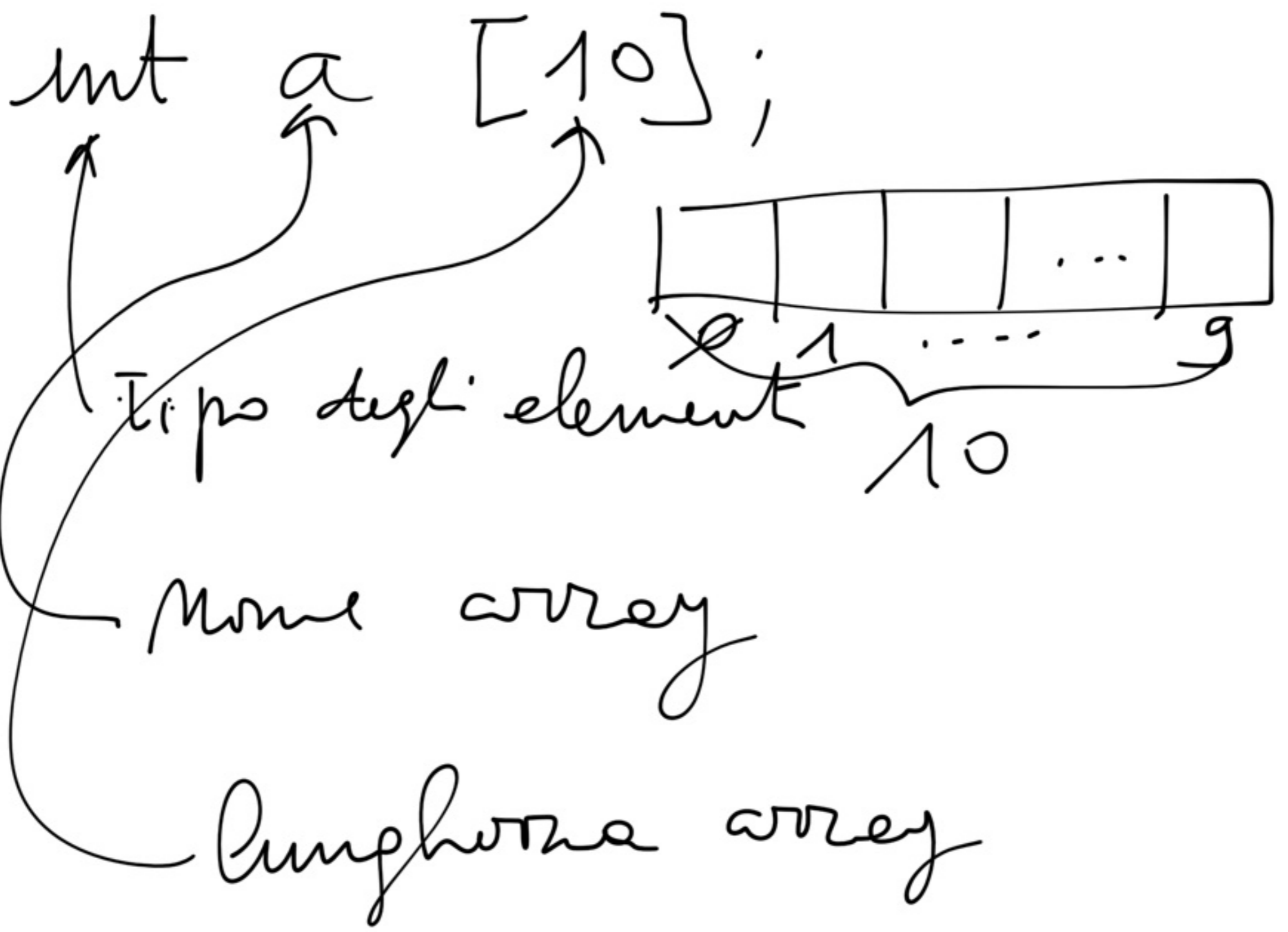


array

sequenza di variabili  
tutte dello stesso  
tipo.



lunghezza =  $m$





int a [10];

a[0] = 5;

a[0] = a[0] + 1;



```
int a[10];  
int i;
```

```
i = 0;
```

```
while (i < 10) {  
    a[i] = 0;  
    i = i + 1;  
}
```

```
...
```

for (i = 0 ; i < 10 ; i = i + 1) {

Annotations:  
- "initialization" points to `i = 0`  
- "controls" points to `i < 10`  
- "increments" points to `i = i + 1`  
- A bracket on the right side of the loop body indicates the loop's extent.

int i;  
int a[10];

for (i = 0 ; i < 10 ; i = i + 1) a[i] = 0;

The entire for loop is circled in red. The increment `i = i + 1` is also circled in black. An arrow labeled `i++` points from the circled increment to the text `i++` above it.