Titolo nota 15/10/2015

Stato come pile di frame (gestire l'esecuzione dei BLOCCHI)

$$\Pi = \{\Omega\} \cup \{f.\pi \mid f: A \to B_\perp \text{ e } \pi \in \Pi\}$$

$$\cdot : (A \to B_\perp) \times \Pi \longrightarrow \Pi$$

$$\pi(a) = \begin{cases} \perp & \text{se } \pi = \Omega \\ f(a) & \text{se } \pi = f.\pi' \text{ e } f(a) \neq \perp \\ \pi'(a) & \text{se } \pi = f.\pi' \text{ e } f(a) = \perp \end{cases}$$

ricorsione

Titolo nota 15/10/2015

$$\pi\left[{}^{b}/_{a}\right]^{add} = f\left[{}^{b}/_{a}\right]^{add} \cdot \pi' \qquad \text{se } \pi = f \cdot \pi' \text{ e } f(a) = \bot$$

$\neq$

$$\pi\left[{}^{b}/_{a}\right]^{mod} = \begin{cases} f\left[{}^{b}/_{a}\right]^{mod} \cdot \pi' & \text{se } \pi = f \cdot \pi' \text{ e } f(a) \neq \bot \\ f \cdot \pi'\left[{}^{b}/_{a}\right]^{mod} & \text{se } \pi = f \cdot \pi' \text{ e } f(a) = \bot \end{cases}$$

Titolo nota 15/10/2015

Funzioni e Procedure ⟶ (Funzioni)

Struttura dei programmi C

includere delle LIBRERIE

Sequenza di dichiarazioni

#include .....

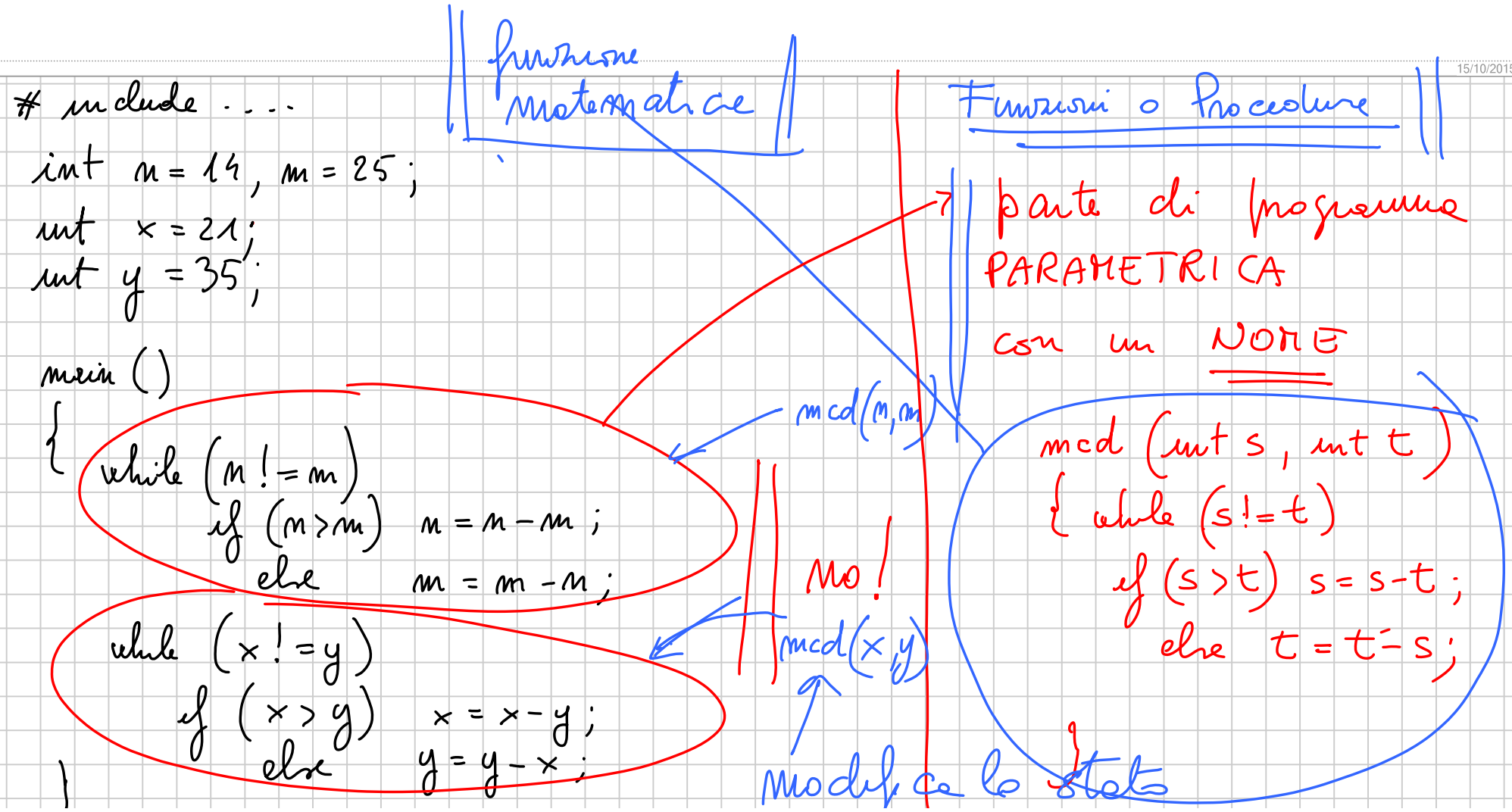
{ dec

main()

{ Blocco

programma principale

Titolo nota 15/10/2015



Funzioni e Procedure in C

Titolo nota 15/10/2015

Possono, oltre a modificare lo stato, restituire un valore.

Dichiarazione di Funzione o Procedura C ha questa struttura

tipo_del_risultato nome_della_f_o_proc (seq. dei parametri con il loro tipo)

{ Blocco

si può avere il comando return

restituisce come risultato della fun. o proc. il valore di Exp

return Exp ;

Titolo nota 15/10/2015

Funzioni

oltre a modificare lo stato restituiscono un valore

tipo_risultato nome_fun (par)

Blocco

deve avere questo tipo

return Exp;

Procedure

Modificano solamente lo stato

void nome_proc (par)

Blocco

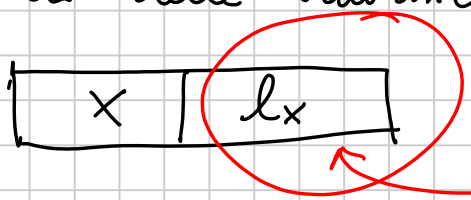
non c'è return

Complicare lo stato:

Titolo nota 15/10/2015

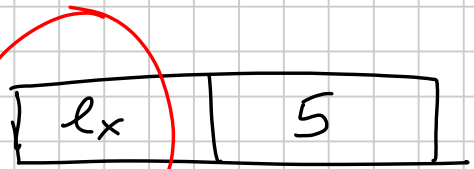
Ambiente

Contiene i nomi delle variabili associati agli indirizzi della memoria (locazioni) dove sono memorizzati i valori delle variabili



Memoria

Associa a degli indirizzi di memoria dei valori (i valori memorizzati a quegli indirizzi)



Titolo nota 15/10/2015

Ambiente
Memoria
} pile di "frame"

Ambiente è una pila di "frame" da Ide in Loc
(locazioni)

Memoria è una pila di "frame" da Loc in Val
(valori)

Useremo ρ per indicare un ambiente
μ per indicare una memoria

Titolo nota 15/10/2015

ρ ambiente. Insieme di tutti gli ambienti ρ minuscolo: P

μ memoria " " tutte le memorie μ minuscolo: M

$$P = \{\Omega\} \cup \{\varphi.\rho \mid \varphi : Ide \to Loc_\perp \text{ e } \rho \in P\}$$

Ω omega maiuscolo — φ fi — ρ ro — P ro maiuscolo

$$M = \{\Omega\} \cup \{\nu.\mu \mid \nu : Loc \to Val_\perp \text{ e } \mu \in M\}$$

M mu maiuscolo — ν nu minuscolo — μ mu minuscolo

N = nu maiuscolo

Come vengono eseguiti i blocchi rispetto al nuovo stato?

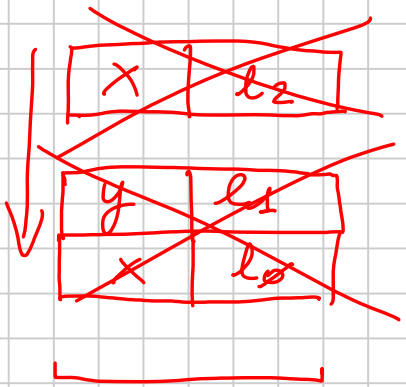
Titolo nota 15/10/2015

Come ci si aspetta!

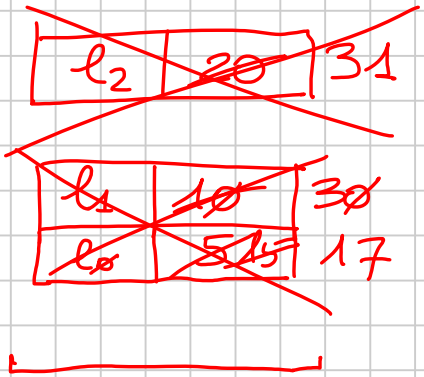
```
{ int x = 5;
  int y = 10;
  x = y + x;
  { int x = 20;
    y = x + y;
    x = y + 1;
  }
  x = x + 2;
}
```

15 10 20 30 15

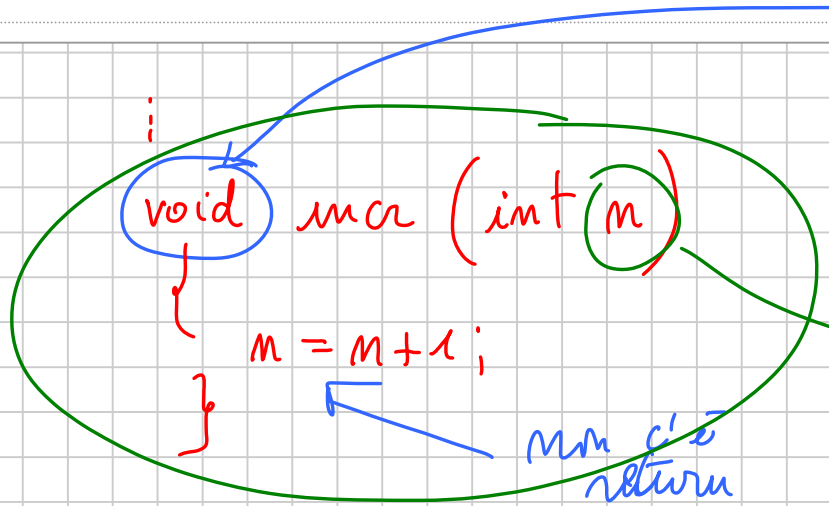
Ambiente



Memoria



Titolo nota 15/10/2015



dichiarazione della procedura

parametro

come si collegano ARGOMENTI e PARAMETRI ?

```
main ()
{
  int z = 10;
  int y = 11;
  incr(z);
  incr(y);
}
```

due usi della procedura (chiamate) sugli argomenti z e y

$f(n) = n + 1$ definizione

$f(3)$ chiamata

= { si sostituisce nelle def. di f al parametro n il valore 3 e si calcola }

$3 + 1$

$= 4$

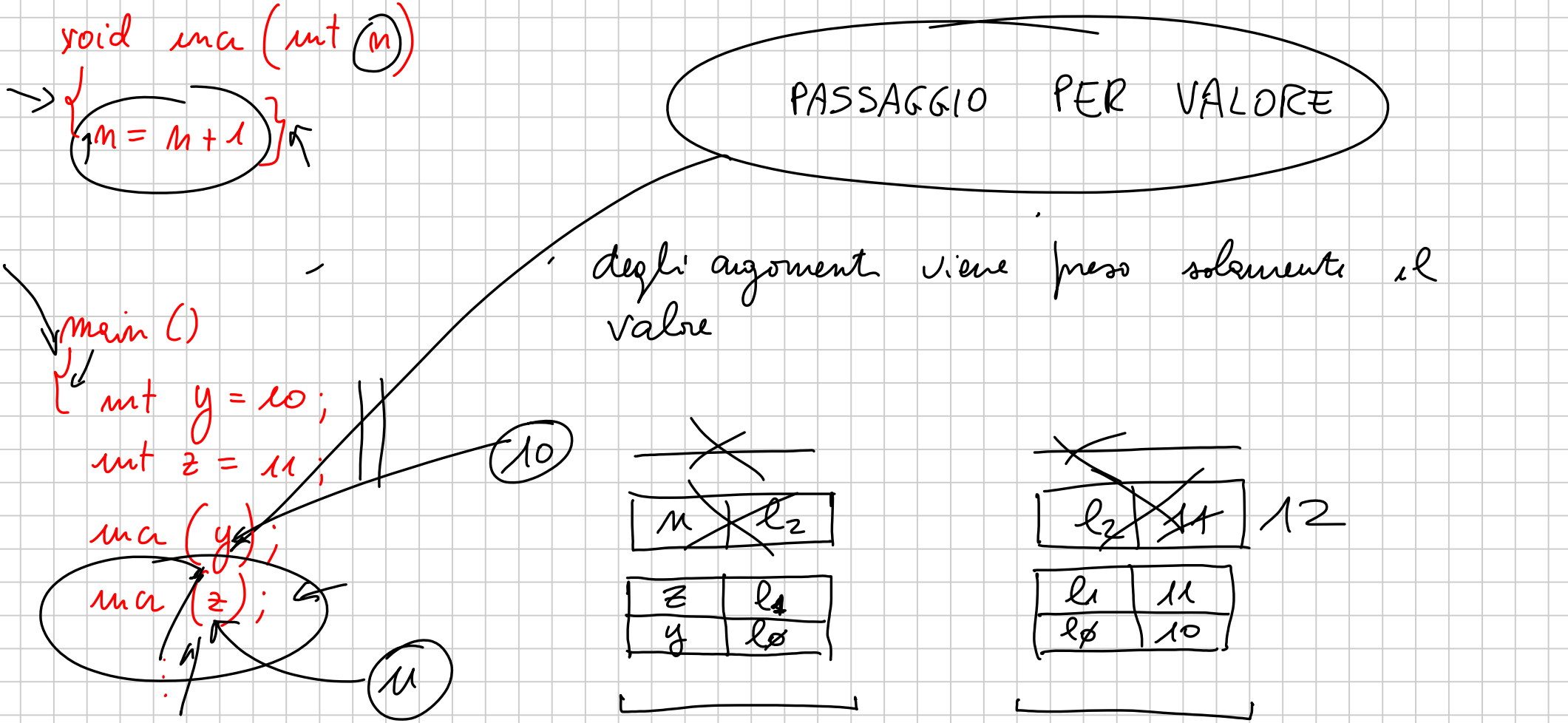
In programmazione imperativa NON SI PUÒ FARE perché tutto deve essere rappresentato da modifiche di STATO e non dei programmi.


```
main()
{ int y = 10;
  int z = 11;
  inca(y);
  inca(z);
}
```

In programmazione imperativa NON SI PUÒ FARE perché tutto deve essere rappresentato da modifiche di STATO e non dei programmi.

- Quando una procedura viene chiamata si aggiunge allo stato un nuovo frame (sia nell'ambiente che nella memoria) che contiene nuove variabili corrispondenti ai nomi dei parametri della procedura.
- Il valore iniziale di queste variabili è il valore degli argomenti della chiamata

PASSAGGIO DEI PARAMETRI "PER VALORE"


void inc (int n)
n = n + 1
PASSAGGIO PER VALORE
degli argomenti viene preso solamente il valore
main ()
int y = 10;
int z = 11;
inc (y);
inc (z);
10
11
n
l2
z
l1
y
l0
l2
11
12
l1
11
l0
10

PUNTATORI (indirizzi di memoria)

abbiamo introdotto anche la memoria per esplicitare gli indirizzi (che in C vengono utilizzati)

```
int inc (int n)
{ n = n+1;
  return n; }

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

funzione

chiamata di funzione deve come risultato un valore.

Quali strutture in C danno come risultato un valore?

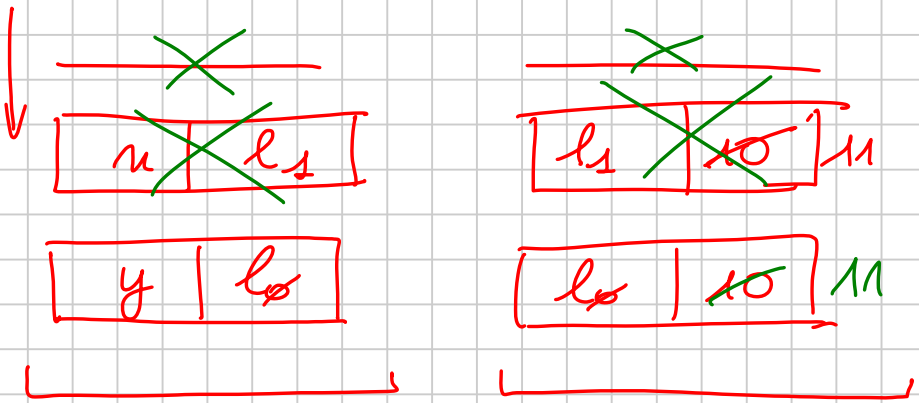
- Exp
- Dec
- Com

```
int inc (int n)
{
  n = n + 1;
  return n;
}
```

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

10

11



puntatori

`int * x = 10;` **errore**

in memoria i valori di x non saranno interi, ma indirizzi di valori interi di memoria

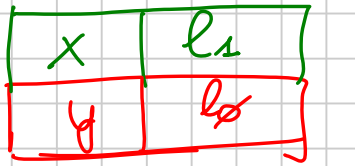
`int * x;`

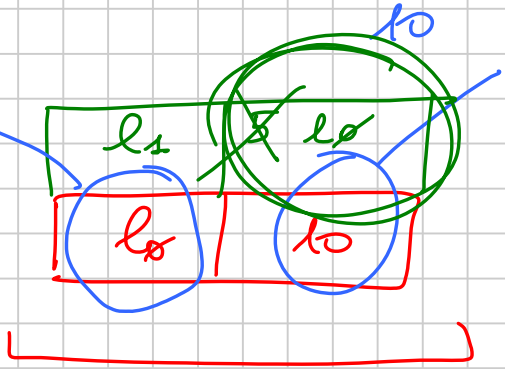
variabile puntatore

`int *` : il tipo degli indirizzi di memoria che possono memorizzare valori interi.

`int` : il tipo dei valori interi

l_0 è un valore di tipo (`int *`) è un indirizzo di una parola di memoria che contiene un valore intero





valore di tipo int