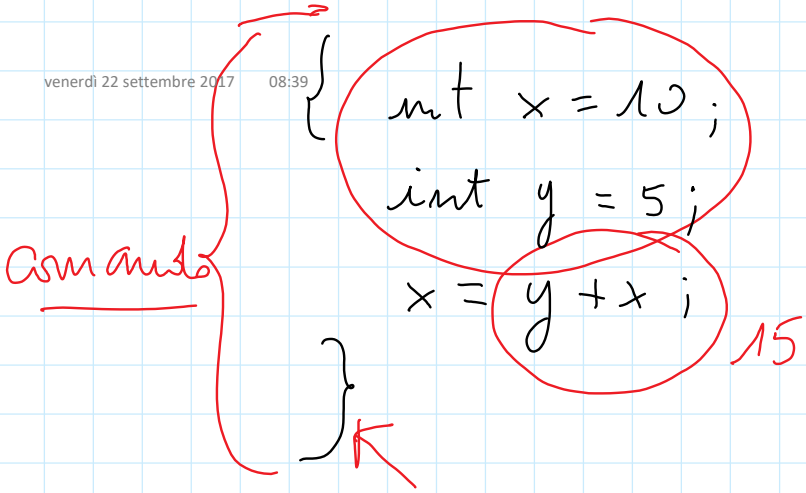
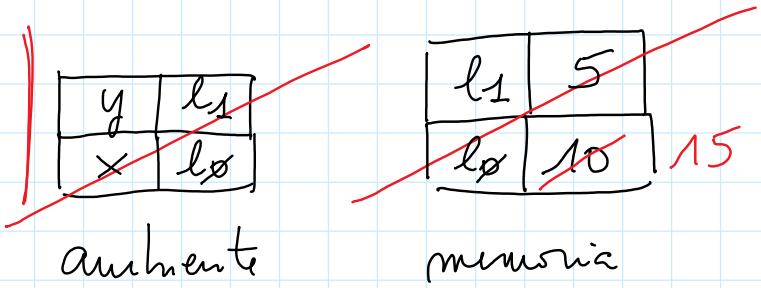
venerdì 22 settembre 2017 08:39

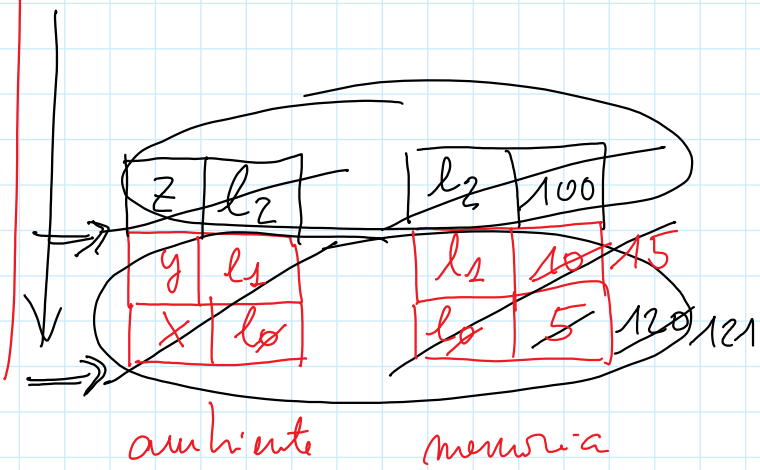




```
{ int x = 5;
  int y = 10;
  y = (y + x;)  15
  { int z = 100;
    x = (z + y + x;)  120
  }
  x = (x + 1;)  121
}
```

Comando

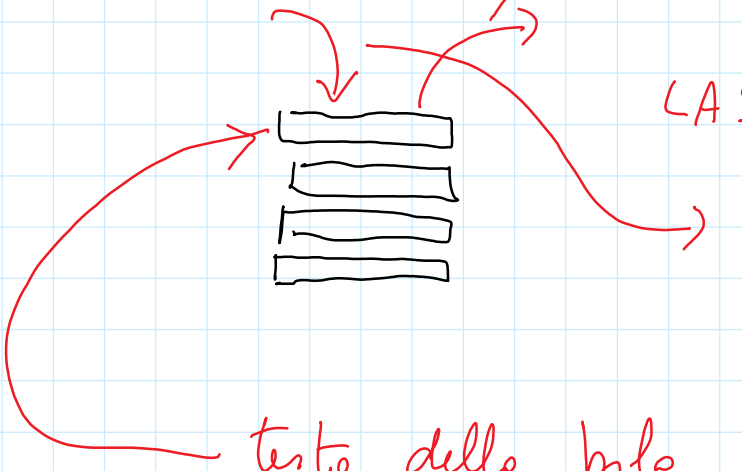
Blocchi annidati



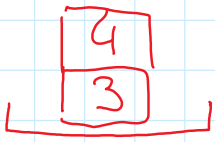
ambiente memoria

venerdì 22 settembre 2017 08:39

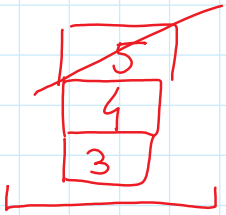
PILA (STACK)



Esiste una pila speciale pila vuota Ω

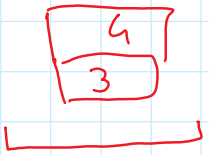


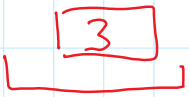
← vuoto



push(p, v)

pop(p)

push( , 4) = 

pop() = 

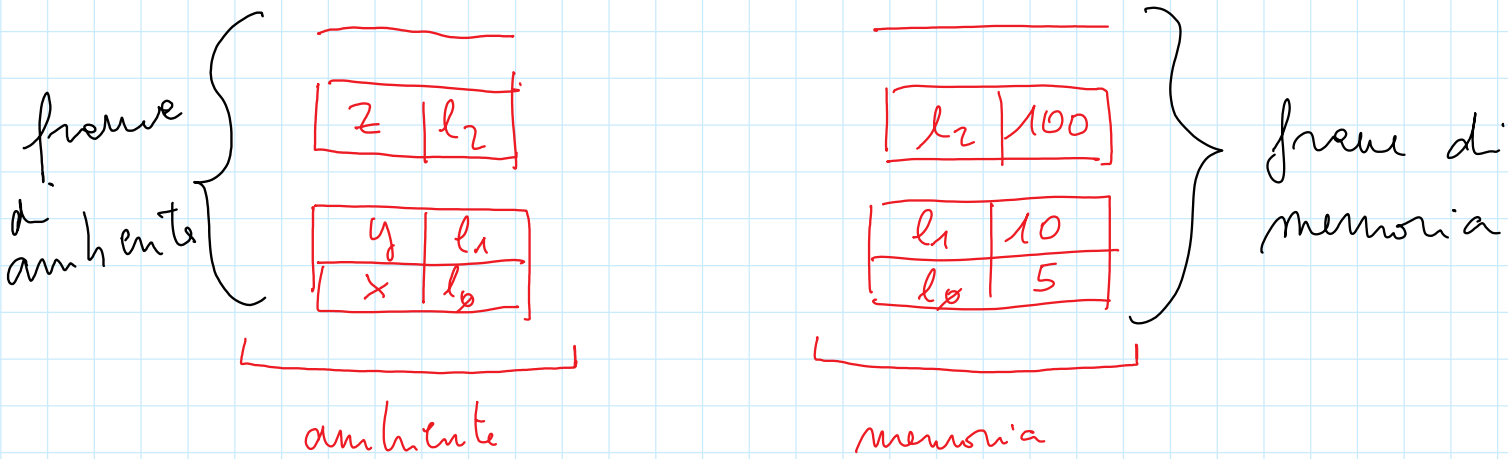
ambiente memoria

"frame" di ambiente "frame" di memoria

Il nuovo stato che consideriamo è costituito da un ambiente che è una PILA di frame di ambiente e da una memoria che è una di frame di memoria

Esiste un "frame" speciale che si chiama frame vuoto e si indica

---


frame di ambiente
z | l2
y | l1
x | l0
l2 | 100
l1 | 10
l0 | 5
frame di memoria
ambiente
memoria

1) Quando si "entra" in un blocco (quando si incontra una graffa aperta) si aggiunge un frame vuoto sulla pila di ambiente e sulla pila di memoria

2) Una dichiarazione crea una associazione nel frame in testa alla pila di ambiente e nel frame in testa alla pila di memoria

3) Quando un blocco termine la sua esecuzione (quando si incontra una graffa chiusa si toglie dalla pila di ambiente e di memoria il frame in testa (POP)

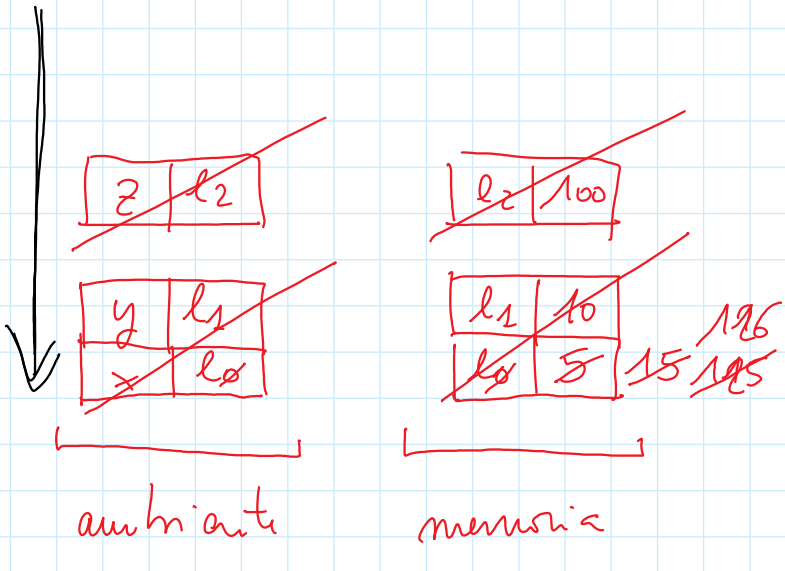
```
{ int x = 5;
  int y = 10;
  x = x + y;
  { int z = 100;
    x = y + x + z;
  }
  x = x + 1
}
```

15

125

126

printf



```
{
    int x = 5;
    int y = 10;
    x = y + x;
    {
        int x = 100;
        y = x + y;
    }
    x = x + 1;
}
```

variabile locale

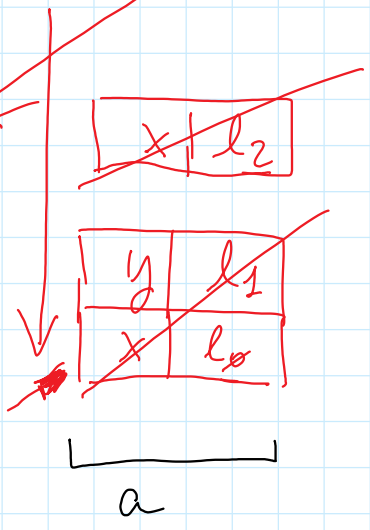
variabile globale

110

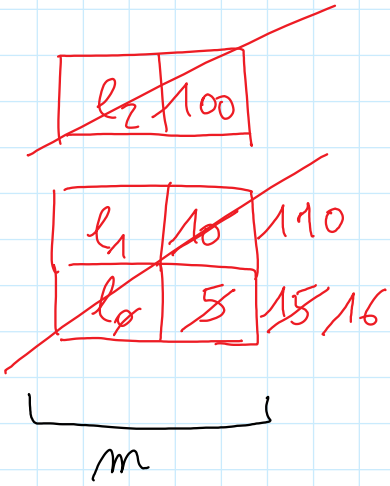
100

10

16



a



m

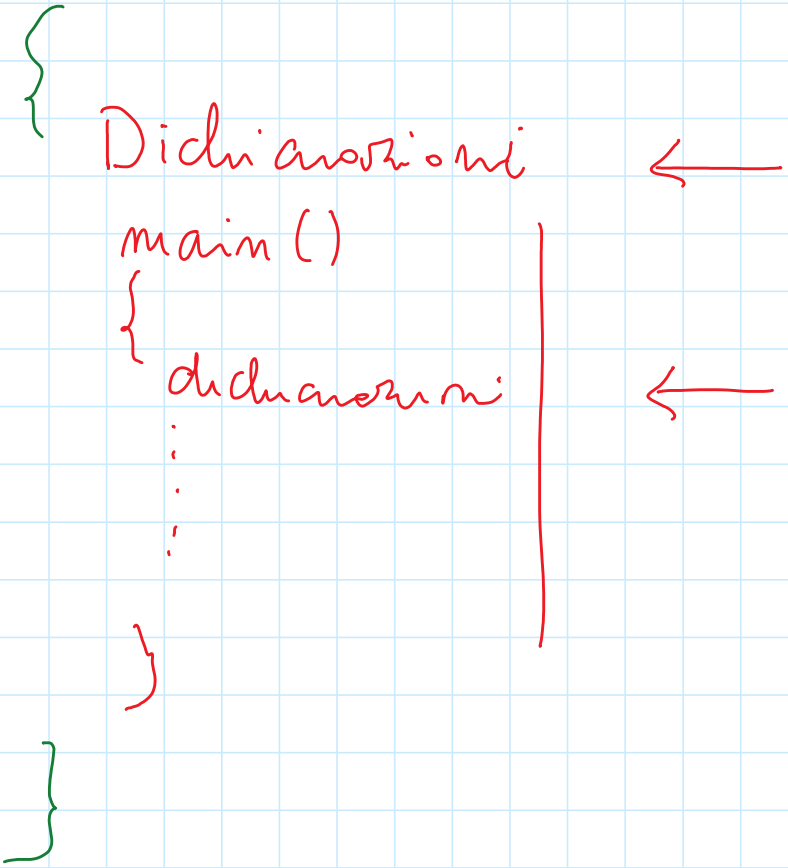
Visibilità degli identificatori:

venerdì 22 settembre 2017 08:39

Una variabile è visibile nel blocco in cui è dichiarata e in tutti i blocchi interni, a meno che non sia ridichiarata in un blocco interno.

STRUTTURA DI UN PROGRAMMA C

venerdì 22 settembre 2017 08:39



venerdì 22 settembre 2017 08:39

```
int n = 45;
int m = 72;
int x = 21;
int y = 15;

main()
{ while (x != y)
    if (x > y)  x = x - y;
    else        y = y - x;
  printf(..., x);
}
```

```
while (n != m)
  if (n > m)  n = n - m;
  else        m = m - n;
```

$f(x, y) = x + y + 3$

definizione di funzione

nome

parametri

espressione che dà il risultato in funzione dei valori dei parametri

---

$f(3, 4) = 10$

chiamata (applicazione)

nome

argomenti (valori)

prendo la def., sostituisco ai parametri gli argomenti, nell'ordine, e valuto l'espressione alla destra dell'uguale nella definizione.

```
void mcd ( int p , int s )
{ while (p != s)
    if (p > s) p = p - s;
    else       s = s - p;
}
```

definizione di PROCEDURA

(definizione)

```
{ int n = 42;
  int m = 75;
  int x = 21;
  int y = 15;
  mcd(n, m);
  mcd(x, y);
  :
```

come si collegano gli argomenti con i parametri ??

chiamata della procedura

esecuzione della procedura

venerdì 22 settembre 2017 08:39

Nella programmazione imperativa il collegamento tra argomenti e parametri di una procedura si chiama:

MODALITÀ DI PASSAGGIO DEI PARAMETRI

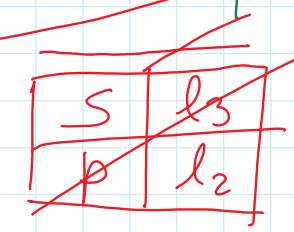
In C esiste una unica modalità di passaggio dei parametri.

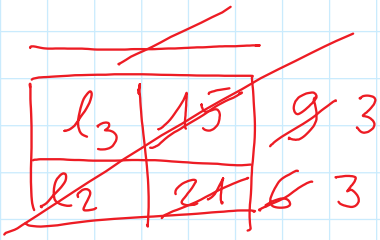
MODALITÀ "PER VALORE"

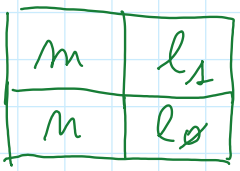
Quando si esegue una procedura (cioè quando chiamata (es. mcd(n,m)) si aggiunge all'ambiente e alla memoria un frame che contiene le variabili con il nome dei parametri (parametri diventano variabili in un nuovo frame). A queste variabili viene dato inizialmente i VALORI degli argomenti.
Al termine dell'esecuzione della procedura il frame viene cancellato.

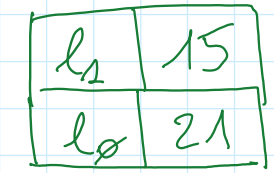
```
void mcd (int p, int s)
{ while (p != s)
    if (p > s)  p = p - s;
    else        s = s - p;
}

main()
{ int n = 21;
  int m = 15;
  mcd (n, m)
}
```

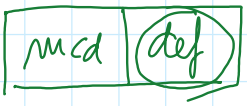
21 15











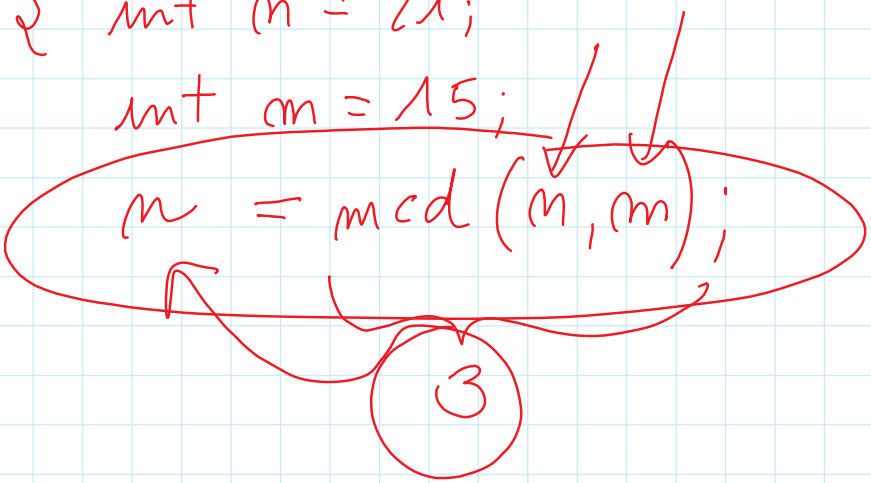
a

m

venerdì 22 settembre 2017 08:39

Comando { Sequenze di dichiarazioni → creare lo stato
Sequenze di comandi → modificare lo stato }

```
int mcd (int p, int s)
{ while (p!=s)
      if.....

   return p;
}
main()
{ int n = 21;
  int m = 15;
  n = mcd(n,m);
```



FUNZIONE

RETURN

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

```
main()
{
int x = fact(4);
}
```