ESERCIZI PER CASA:

```
{int a=0; y=1;
 (*)
 void Exec(){ int x;
                (**)
                y=fun();
                a=a+1;
              }
 While (a<=1){ exec();}
```

① Cosa devo definire: fun()

x Non è Inizializzata Ma Nemmeno Usata Quindi è OK

Dobbiamo Utilizzare una Variabile NON Locale che Abbia sia un Ambiente a Tempo di definizione Sia Nel Blocco di Chiamata

Bisogna Sempre Giustificare!

Se possibile MAI Usare la Variabile del CICLO

Salgo x Visto che NON ha Ancora Nessuno SCOPO Nel Codice:

BLOCCO DI DEFINIZIONE (*)

```
int x = 10;
int fun(){return x;}
```

RIFERIMENTO AD UNA VARIABILE NON GLOBALE, Quindi devo sfruttare le REGOLE DI SCOPING

BLOCCO DI CHIAMATA (**)

```
x = a;
```

Deve Cambiare Nelle Esecuzioni, Quindi Sfrutto la Variabile del Ciclo

SCOPING STATICO: fun() ha come AMBIENTE Quello Globale e Restituisce Sempre 10

SCOPING DINAMICO: fun() ha Come AMBIENTE Quello dentro EXEC dove x=a per Ogni Chiamata e Quindi

- Vale 0 alla prima Chiamata
- Vale 1 alla SECONDA

Catena Statica/Dinamica un Altro Esercizio possibile

C'è Sempre la DEF. di Scoping Statico & Dinamico e la differenza

BINDING:

- **DEEP**: Si Riferisce all' AMBIENTE dove si CREA il legame
- **SHALLOW**: Si Riferisce all' AMBIENTE Quando Viene Effettuata la Chiamata

Quando Si parla di BINDING? Definire lo SCOPING. (Statico / Dinamico)

Quando una PROCEDURA ha un AMBIENTE non locale, Servono Anche Quelle di BINDING Quando la procedura con AMBIENTE NON locale è passata Come PARAMETRO ad un Altra PROCEDURA

```
int x=3, y=2;                                  (1)
int fun1(int n){ return x+y+n;}

int fun2(int h(int B)){
    int x=4; y=4;                              (3) AMBIENTE di Chiamata con PARAMETRO ATTUALE
    return h(5)+x
}
:
{ int x=6, y=3; z= fun2(fun1);
```

Mi Serve Scoping

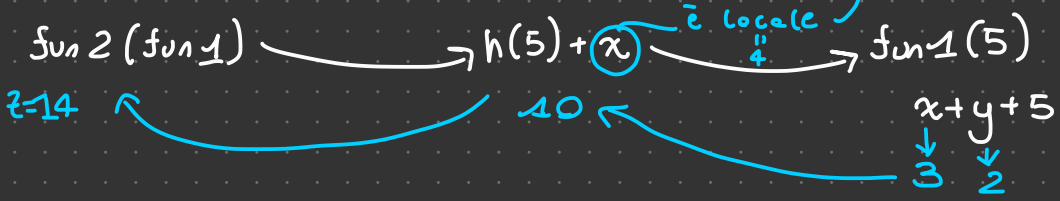
Mi Serve BINDING

(2) AMBIENTE dove viene CREATO il Legame

① Ambiente di Definizione

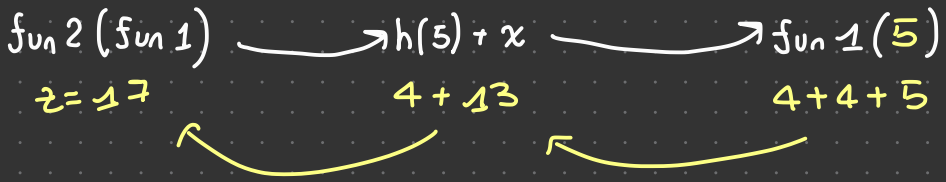
Procediamo per casi:

Scoping Statico Viene restituito $z=14$; visto che ho $x=3$ e $y=2$ (Ambiente ①)

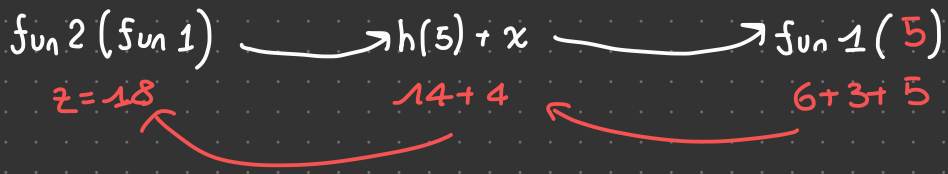


Scoping Dinamico:

① Shallow ⇒ $z=17$; (Ambiente ③)



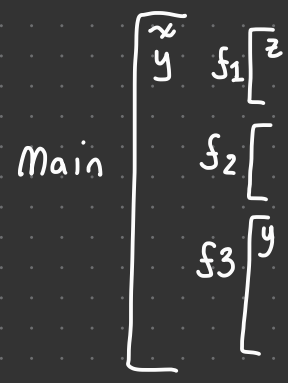
② Deep ⇒ $z=18$; (Ambiente ②)



SCOPING IMPLEMENTAZIONE:

Sempre DEFINIRE Concetti di SCOPING, Cos'è Link Statico e anche la CRT (definizione e com'è Aggiornata/fatta)

```
int x,y;
void f1(){int z=x+y;}
void f2(){ y=y+1}
void f3(){ int y=x+2;
           f2();
           f1();
         }
```



x=5, y=2;
fun3();

Sd(Main) = 0
Sd(f1) = Sd(f2) = Sd(f3) = 1

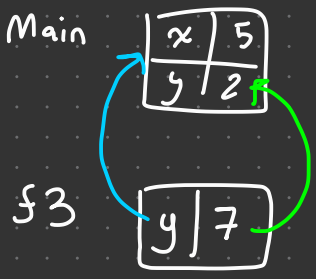
CATENA DELLE CHIAMATE ⇒ f3 → f2 → f1

Scoping Statico:

Main

x	5
y	2

Chiamata a f3()



$K_{f_3} = Sd(\text{chiamante}) - Sd(\text{chiamato}) + 1 = Sd(Main) - Sd(f3) + 1 = 0$

$CS(f3) = Main$

$y = x + 2 = 7$

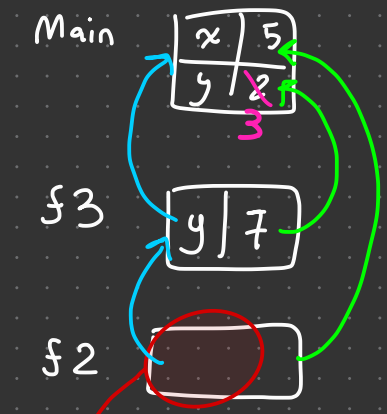
Devo Risolverla:

$x = 5$

$N_x = Sd(f3) - Sd(Main) = 1$

sd(chi la usa) - Sd(più vicina che la def.)

Chiamata f3 a f2



$K_{f_2} = Sd(f3) - Sd(f2) + 1 = 1$

$CS(f2) = CS(f3) = Main$

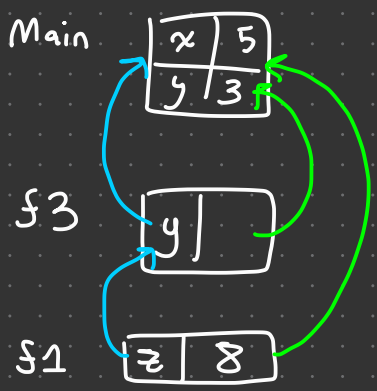
$y = y + 1 = 3$

non è locale:

$N_y = Sd(f3) - Sd(main) = 1$

Risalgo di 1 Volta la Catena Statica

Ci vanno Solo Quelle DICHIARATE e BASTA !!!



$K_{S_1} = Sd(f_3) - Sd(f_1) + 1 = 1$

$CS(f_1) \neq CS(f_2) = Main$

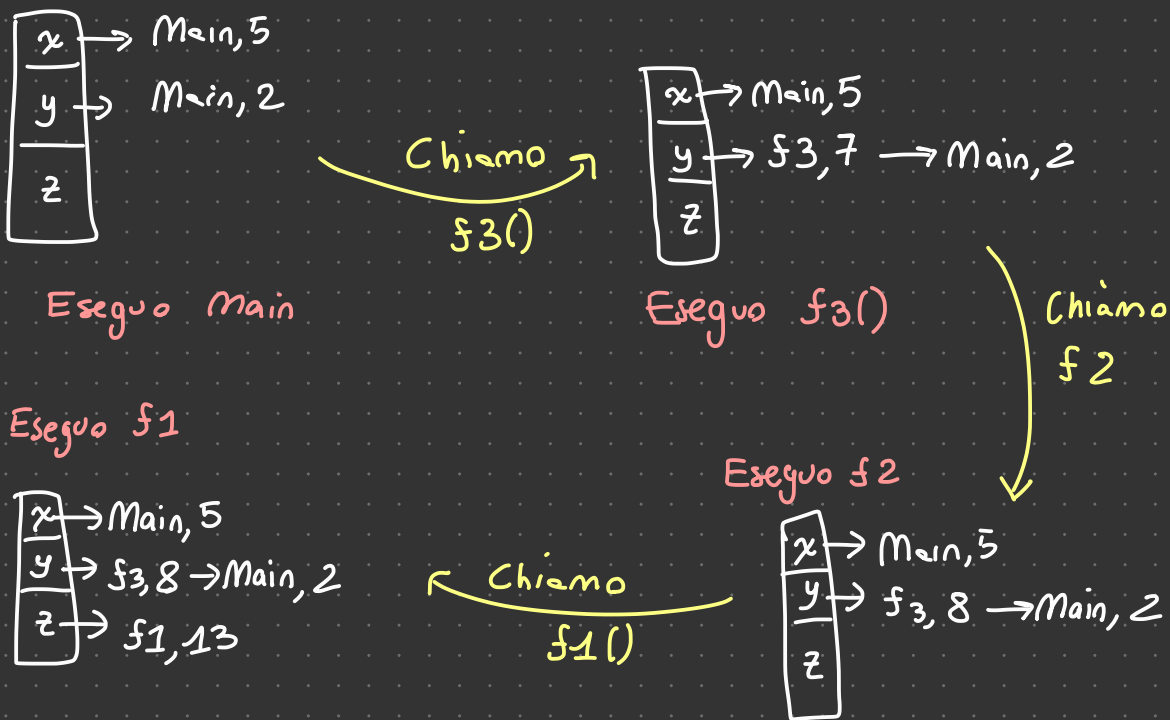
$z = x + y = 8$

$N_y = Sd(f_1) - Sd(Main) = 1$

$N_x = Sd(f_1) - Sd(Main) = 1$

Poi via via si distruggono le Chiamate

Scoping Dinamico:



Induzione Naturale NON c'è Ma c'è Solo Quelle STRUTTURALE

PASSAGGIO PER VALORE:

Def. Tipologia di Passaggio dell' Esercizio

Risultato NON fatto!

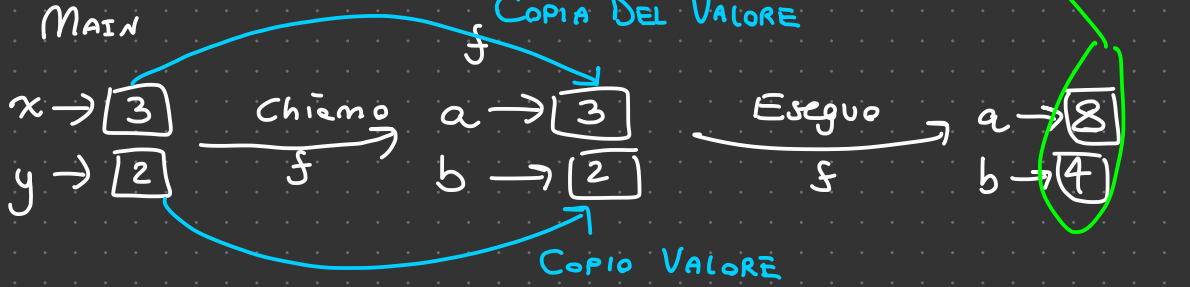
Codice che definisce una funzione e UTILIZZO:

```
f(int a, int b){
   a=a+5;
   b=2b;
  }

main(){
  int x=3, y=2;
  f(x,y);
}
```

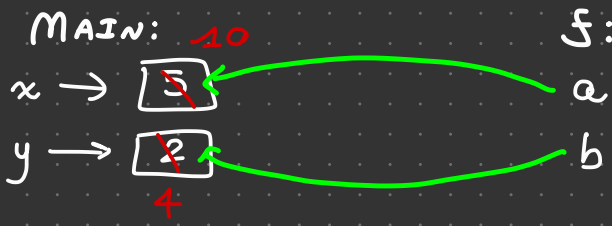
Cosa Succede in MEMORIA?

SOVRASCRITTE ALCE VAR DI PRIMA (Stesse Loc)



Sia a,b Vengono Eliminate Mentre x,y Rimangono INVARIATI

PASSAGGIO PER RIFERIMENTO:



NON Creo Spazio con Valori ma Avrà INDIRIZZI in Modo che puntino alla Stessa Cella di x,y

Si CONCLUDE il MAIN con le Variabili x e y Con NUOVI Valori visto che la MODIFICA Viene Osservata/ha Valore anche All'esterno della PROCEDURA (Una Volta che ho Eliminato Gli Indirizzi.

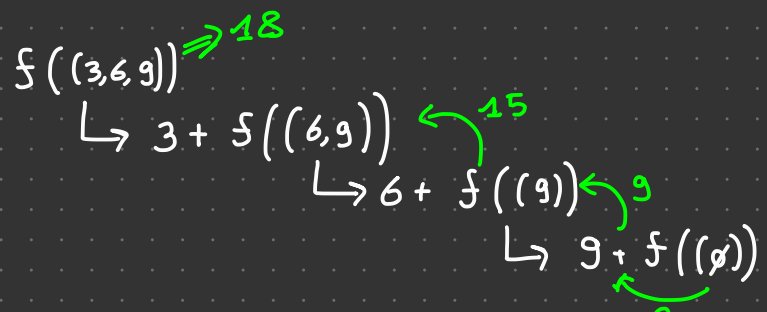
RICORSIONE:

Da RICORSIVO, RICORSIVO IN CODA (da DEFINIRE ENTRAMBI)

Mostrare Anche le Sequenze delle Chiamate oltre a dire che Cosa fa

```
int f(lista list){
    IF len(list)==0 return list;
    ELSE return (car(list) + function(cdr(list))
```

SU INPUT (3,6,9)



Questa f.z. Ricorsiva fa la somma di Tutti Gli Elementi della lista

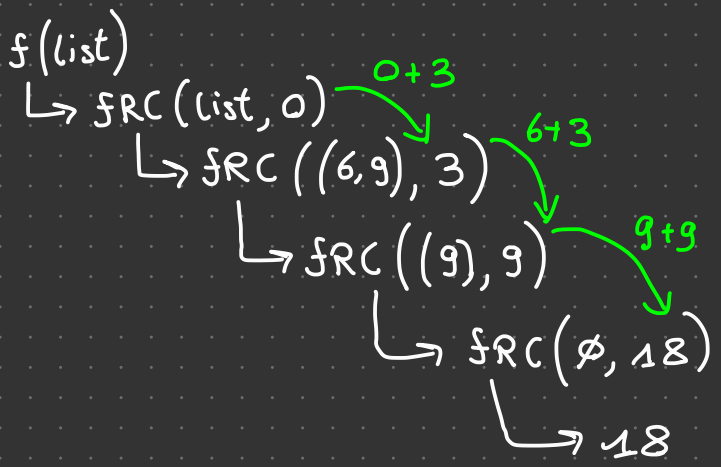
```
int f(lista list)
    return fRC(list,0)
```

Ricordarsi di far PARTIRE Ricorsione di Coda

Risultato che MAN MANO mi Costruisco

```
int fRC(lista list, int Res)
   IF len(list)==0
        return Res;
   ELSE return fRC(cdr(list, res + car(list))
```

Spesso il Valore del Caso BASE è il Valore al Quale Viene Inizializzato il Valore del RISULTATO PARZIALE che Costruirò e dovrò RITORNARLO al POSTO del Valore Costante.



SEMANTICA:

$\rho = [x \mapsto 2, y \mapsto l_y]$ → Va detto cos'è Un AMBIENTE

Associa ID ↦ Val

$\delta = [l_y \mapsto 4]$ → def. Mem

ASSEGNAMENTO: $y := (x+4) * y$

① Valuto $(x+4)$ Quindi RECUPERO il Valore di x dall'Ambiente

$$\frac{\rho \vdash \langle x+4, \delta \rangle \rightarrow^* 6, \delta}{\rho \vdash \langle (x+4) * y, \delta \rangle \rightarrow \langle 6 * y, \delta \rangle} \quad \rho(x) = 2$$

② Valuto y:

$$\frac{\langle \rho \vdash 6 * y, \delta \rangle \rightarrow^* \langle 6 * 4, \delta \rangle}{\rho \vdash \langle 6 * y, \delta \rangle \rightarrow \langle 24, \delta \rangle} \quad \rho(l_y) \text{ e } \delta[l_y \mapsto 4]$$

③ Eseguo Moltiplicazione

$$\frac{\rho \vdash \langle 6 * y, \delta \rangle \rightarrow \langle 24, \delta \rangle}{\rho \vdash \langle 6 * 4, \delta \rangle \rightarrow}$$

④ Completo Assegnamento Con Elaborazioni Precedenti

$$\frac{}{\rho \vdash \langle y := (x+4) * y, \delta \rangle \rightarrow^* \langle y := 24, \delta \rangle} \Rightarrow \delta[l_y \mapsto 24]$$

SEMANDICA STATICA STESSO COMANDO:

$\Delta = [x \mapsto int, y \mapsto int_{loc}]$

$$\frac{\Delta \vdash (x+4) * y : int}{\Delta \vdash y := (x+4) * y : int} \quad , \Delta(y) = Int_{Loc}$$

$$\frac{\Delta \vdash x : int \qquad \vdash 4 : int}{\Delta \vdash x+4 : int} \; \Delta(x) = int$$

Assioma

$$\frac{}{\Delta \vdash y : int} \; \Delta(y) = int_{Loc}$$

Quindi c'è COERENZA di TIPO

Ca MOLTIPLICAZIONE tra 2 Tipi INTERI Rimane INTERA, Quindi Posso Affermare che il Comando è Ben formato