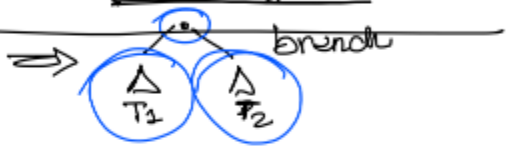
INDUZIONE STRUTTURALE

Definiamo BTree come

- foglia ∈ BTree
- $T_1 \in BTree$ e $T_2 \in BTree \Rightarrow branch(T_1, T_2) \in BTree$



Definiamo $m : BTree \to \mathbb{N}$ conta i nodi di un albero binario

$$\begin{cases} m(foglia) = 1 \\ m(branch(T_1, T_2)) = 1 + m(T_1) + m(T_2) \end{cases}$$

Definiamo $h : BTree \to \mathbb{N}$ calcola l'altezza dell'albero

$$\begin{cases} h(foglia) = 0 \\ h(branch(T_1, T_2)) = 1 + \max(h(T_1), h(T_2)) \end{cases}$$

calcola il valore massimo tra due valori

Dimostrare per induzione strutturale che

(*) $\forall T \in BTree.\ m(T) \leq 2^{h(T)+1} - 1$

Base: Dobbiamo dimostrare (*) per gli elementi foglia

$T = foglia$ $\quad m(T) = m(foglia) = 1$

$h(T) = h(foglia) = 0$

Calcoliamo $2^{h(T)+1} - 1 = 2^{0+1} - 1 = 1$

$$n(T) = 2^{h(T)+1} - 1 \quad \checkmark$$

Passo induttivo: $T = \text{branch}(T_1, T_2)$

Supponiamo per Hp. ind. che

Hp. ind $\rightarrow$
- ⊛ $n(T_1) \leq 2^{h(T_1)+1} - 1$
- ⊛ $n(T_2) \leq 2^{h(T_2)+1} - 1$

dimostriamo che $n(T) \leq 2^{h(T)+1} - 1$

$n(T) = 1 + n(T_1) + n(T_2)$ $\quad h(T) = 1 + \max(h(T_1), h(T_2))$

$= h$

$n(T) = 1 + n(T_1) + n(T_2) \leq 1 + 2^{h(T_1)+1} - 1 + 2^{h(T_2)+1} - 1$

per def di n

per ⊛ ⊛

Hp. ind $\rightarrow 2(2^h) = 2^{h+1}$

$\leq 2^h + 2^h - 1$

$= 2^{h+1} - 1$

$= 2^{h(T)+1} - 1$

$h \geq h(T_1) + 1$

$h \geq h(T_2) + 1$

BTree, definiamo

$\ell : \text{BTree} \rightarrow \mathbb{N}$ conta le foglie

$\begin{cases} \ell(\text{foglia}) = 1 \\ \ell(\text{branch}(T_1, T_2)) = \ell(T_1) + \ell(T_2) \end{cases}$

Dimostrare che $\ell(T) \leq 2^{h(T)} \quad \forall T \in \text{BTree}$

Base: $T = \text{foglia}$: $\ell(T) = 1 \quad 1 \leq 2^0 = 1$ ✓

$h(T) = 0$

Passo induttivo $T = \text{branch}(T_1, T_2)$

Hp. ind: $\ell(T_1) \leq 2^{h(T_1)} \quad \ell(T_2) \leq 2^{h(T_2)}$

Dobbiamo dimostrare che $\ell(T) \leq 2^{h(T)}$

$\ell(T) = \ell(T_1) + \ell(T_2) \quad h(T) = 1 + \max(h(T_1), h(T_2))$

h

$\ell(T) = \ell(T_1) + \ell(T_2) \leq 2^{h(T_1)} + 2^{h(T_2)}$

per def di ℓ — Hp. ind.

$h = \max(h(T_1), h(T_2))$

$\Rightarrow$

$h(T_1) \leq h$

$h(T_2) \leq h$

$\leq 2^h + 2^h = 2(2^h) = 2^{h+1} = 2^{h(T)}$

Exp (espressioni aritmetiche) definite induttivamente

$$\begin{cases} m \in Exp \\ e_1, e_2 \in Exp \;.\; e_1 + e_2 \in Exp \quad e_1 - e_2 \in Exp \end{cases}$$

def. di exp

Def. $val : Exp \to \mathbb{N}$ numero di numeri m presenti in una exp

$$\begin{cases} val(m) = 1 \\ val(e_1 + e_2) = val(e_1 - e_2) = val(e_1) + val(e_2) \end{cases}$$

Def $op : Exp \to \mathbb{N}$ numero di operatori in exp $(+, -)$

$$\begin{cases} op(m) = 0 \\ op(e_1 + e_2) = op(e_1 - e_2) = op(e_1) + op(e_2) + 1 \end{cases}$$

Dimostrare che $val(e) = op(e) + 1 \quad \forall e \in Exp$

Base: $e = n$ $\quad val(e) = val(n) = 1$

$op(e) = op(n) = 0 \quad 1 = 0 + 1$ ✓

Passo ind: $e_1 + e_2$ $\quad (e_1 - e_2$ analogo)

perché la definizione di op e val non dipende dal tipo di operazione

Hp. ind. $\quad val(e_1) = op(e_1) + 1$

$val(e_2) = op(e_2) + 1$

Dobbiamo dim. che $val(e) = op(e) + 1$

$val(e) = \underbrace{val(e_1) + val(e_2)}_{\text{Hp. ind.}} = op(e_1) + 1 + op(e_2) + 1$

↑ per def di val

$= \underbrace{op(e_1) + op(e_2) + 1}_{op(e)} + 1 = op(e) + 1$

per def di op

```
{ int x=1;
  int y=2;
  void A(){ int z= x+y;}
  void B(){ int x=2;
            int w=4;
            void C(){ int x= y+w;
                      A();
                    }
            y= x+y;
```

main { x, y, A{ z }, B{ x, w, C{ x } } }

Sd(main) = 0

Sd(A) = Sd(B) = 1

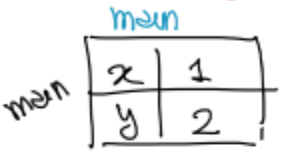
Sd(C) = 2

}

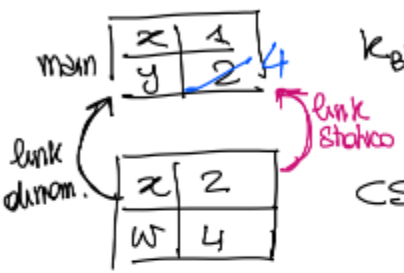
B();
}

main → B → C → A

Scoping statico



Chiamata main → B



chiamante chiamato

$k_B = Sd(main) - Sd(B) + 1$

$= 0 - 1 + 1 = 0$

$CS(B) = main$

Esecuzione di B

locale = 2

$y = x + y$

↳ $N_y = -Sd(main) + Sd(B)$

la più vicina nella colonna di annidamento statico che def. y

$\Rightarrow y_{main} = 2$

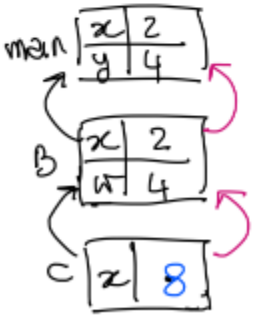
$= 0 + 1 = 1$

uso y

$y = 2 + 2 = 4$ nella main

Dobbiamo risalire di 1 la catena statico per trovare il corretto riferimento a y

Chiamata B → C



Link statico:

$k_C = Sd(B) - Sd(C) + 1$

$= 1 - 2 + 1 = 0$

$CS(C) = B$

locale

$z = y + w$

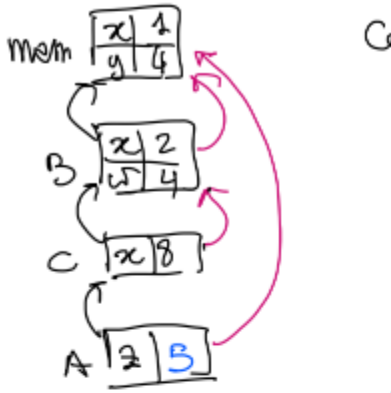
$N_w = Sd(C) - Sd(B)$

$= 1$ in B → 4

8

$N_y = Sd(C) - Sd(main) = 2$ in main

Chiamata C → A



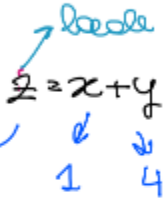
Catena statica

$K_A = Sd(C) - Sd(A) + 1$

$= 2$

$CS(A) = CS(CS(C)) =$

$= CS(B) = main$

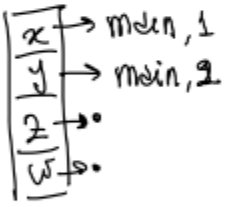


$Ny = Nz = Sd(A) - Sd(main)$

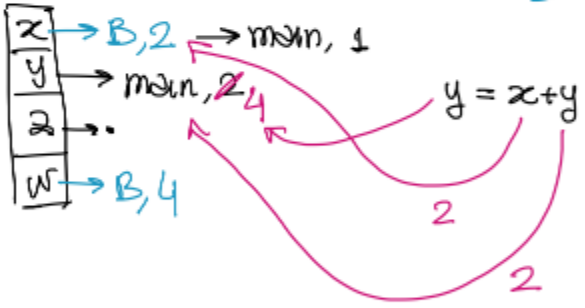
$= 1 \rightarrow main$

Scoping dinamico

CRT



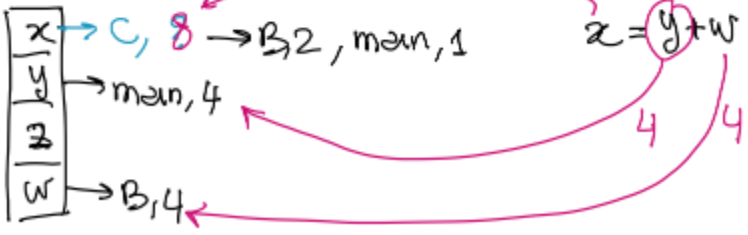
Chiamata main → B



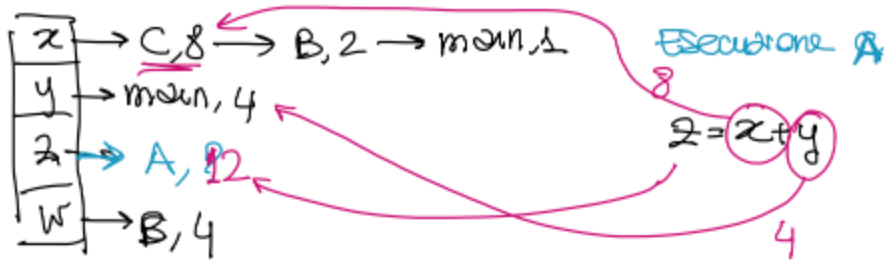
Esecuzione B

Chiamata B → C

Esecuzione C



Chiamata C → A



RICORSIONE

```
lista function (lista list){
    if (length(list)=0) then return list = ()
    else return concat (function (cdr (list)), car (list))
}
```

function(3, 6, 9)

↳ concat (function (6,9), (3)) (9,6,3)

↳ concat (function ((9)), 6) (9,6)

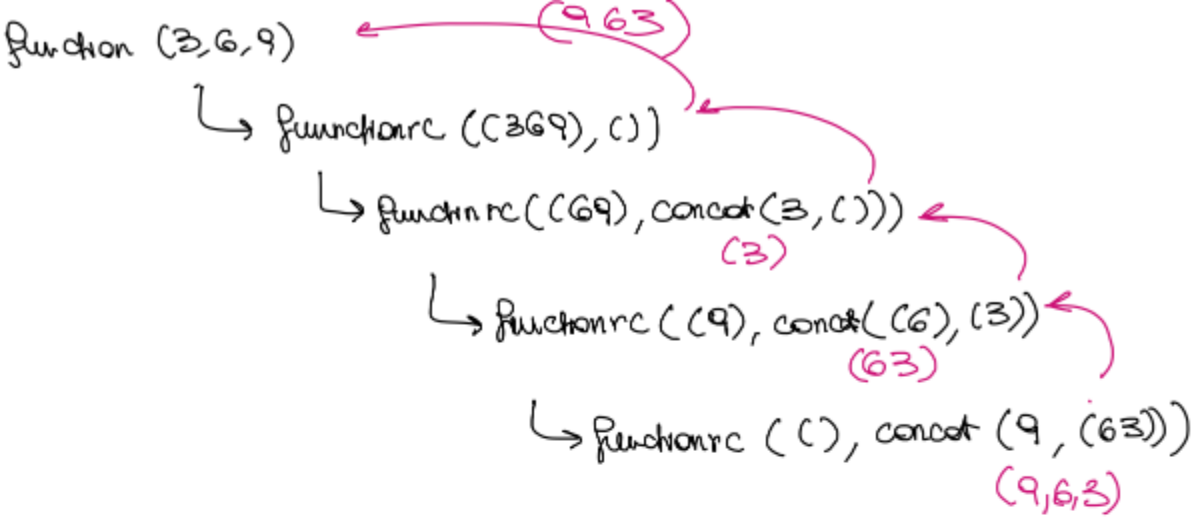
↳ concat (function (()), 9) (9)

()

la funzione inverte gli elementi della lista

```
lista functionrc (lista list, lista res) {
    if length(list)=0 return res;
    else return functionrc (cdr (list), concat (car(list), res));
}
```

```
lista function (lista list) { return functionrc (list, ()); }
```


function (3,6,9)
963
function rc ((369), ())
function rc ((69), concat (3, ()))
(3)
function rc ((9), concat ((6), (3))
(63)
function rc ((), concat (9, (63)))
(9,6,3)