SCOPING

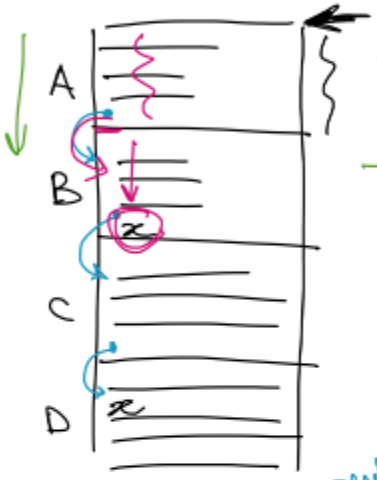
STATICO

Costruisce la catena statica

DINAMICO

CRT (Tabella centrale di Riferimenti)

⇒ mantiene aggiornata l'associazione tra oggetto e ambiente



RdA in esecuzione

→ Shallow Access

CRT

→ Un array di elementi indicizzati sul nome delle variabili

→ Ogni entry è una lista linkata di elementi, ogni elemento "descrive" un ambiente in cui la variabile è definita

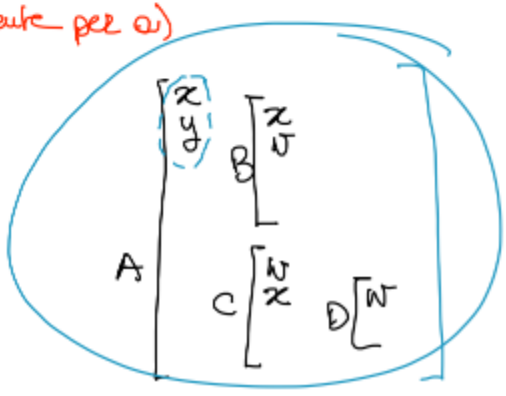
ambienti in cui x è definito e che sono ancora attivi

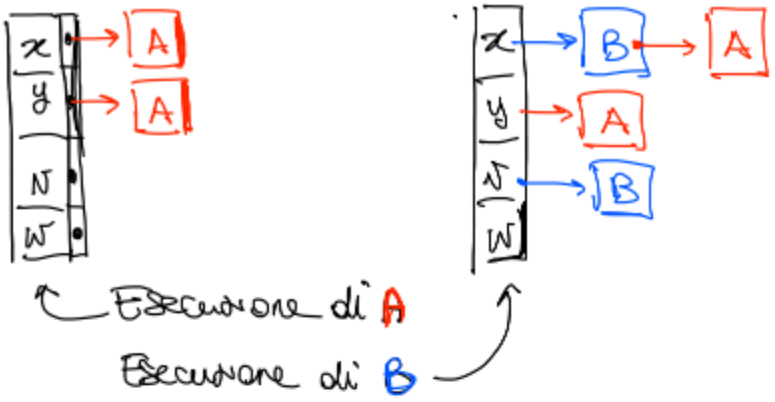
è l'ultimo ambiente attivato durante l'esecuzione

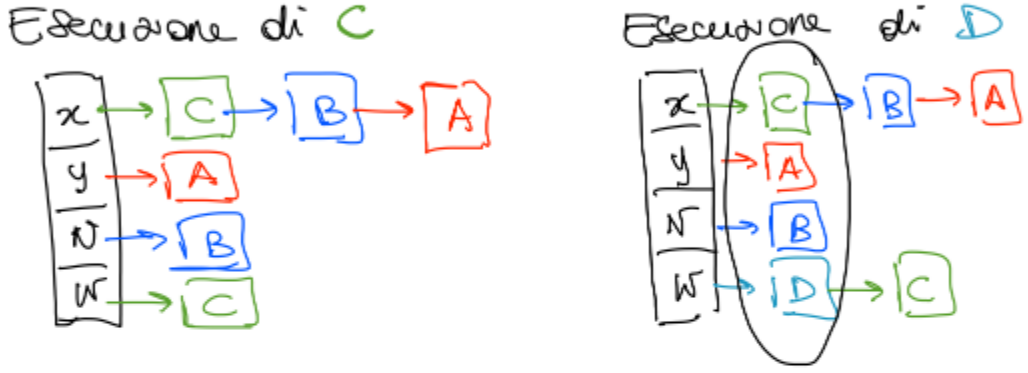
⇒ tempo di accesso costante

procedure (ambiente per a) che termina

A → B → C → D



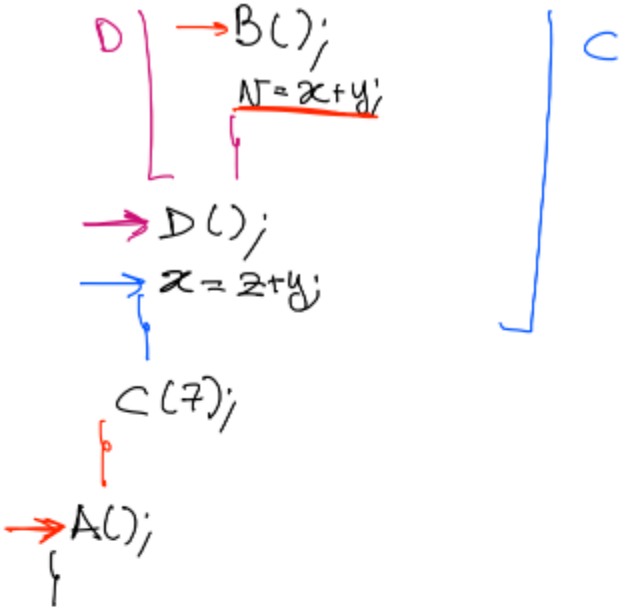


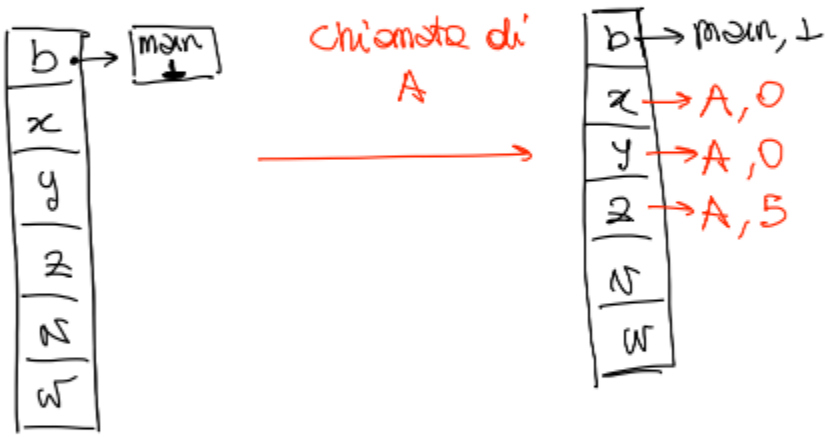


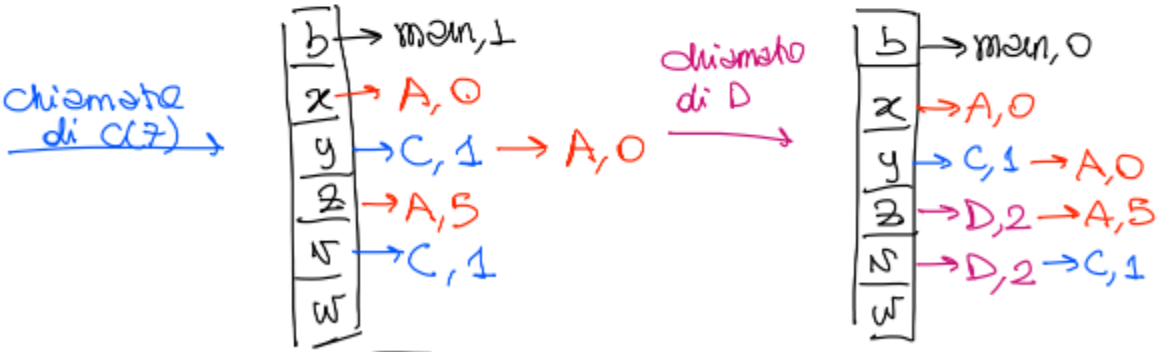
Esempio

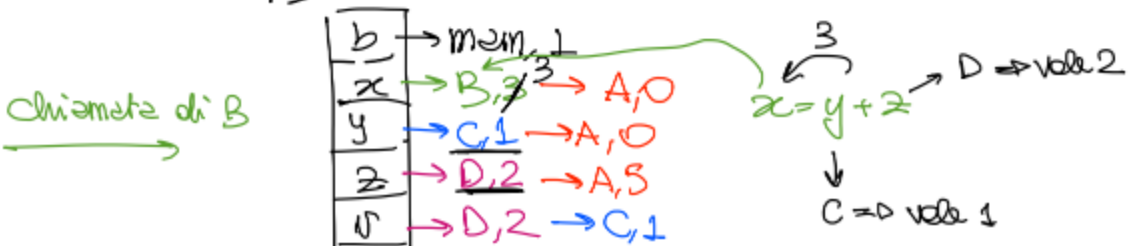
```
{ int b;
  Void A(){
        int x=0, y=0, z=5;
        Void B(){
            int x=3, W=3;
        →   x= y+ z;
        }
        void C(int a){
            int y=1, N=1;
          [ void D(){
```

B


D
B();
N = x+y;
C
D();
x = z+y;
C(7);
A();

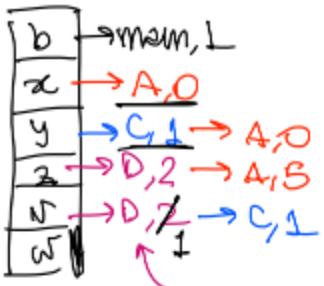

b
x
y
z
v
w
main
Chiamata di A
b → main, 1
x → A, 0
y → A, 0
z → A, 5
v
w

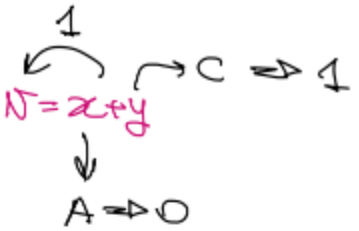

Chiamata di C(7)
b → main, 1
x → A, 0
y → C, 1 → A, 0
z → A, 5
v → C, 1
w
Chiamata di D
b → main, 0
x → A, 0
y → C, 1 → A, 0
z → D, 2 → A, 5
v → D, 2 → C, 1
w


Chiamata di B
b → main, 1
x → B, 3 → A, 0
y → C, 1 → A, 0
z → D, 2 → A, 5
v → D, 2 → C, 1
w
3
x = y + z
D ⇒ vale 2
C ⇒ vale 1

Termina B

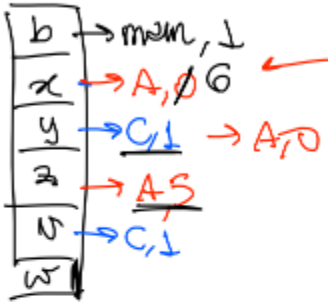
Siamo in D

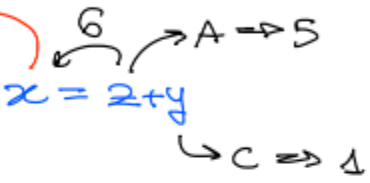




Termina D

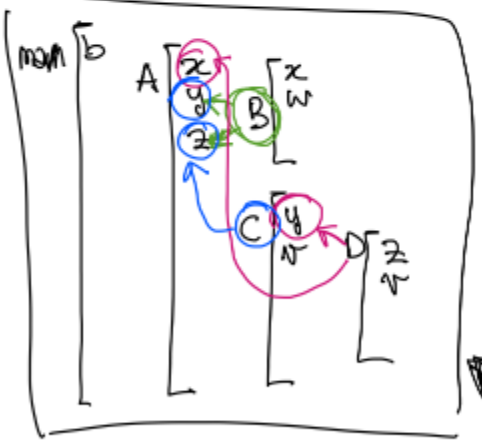
Siamo in C





Il resto del codice termina la chiamata e distruppe via via le fisk della CRT

Scoping statico



Sol (main) = 0

Sol (A) = 1

Sol (C) = Sol (B) = 2

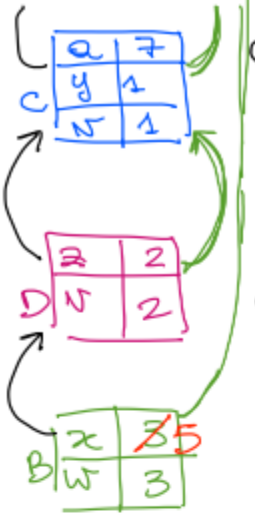
Sol (D) = 3

Esecuzione main

Chiamata di A

Link statico : $R_A = Sol(main) - Sol(A) + 1$

$= 0 - 1 + 1 = 0$



Chiamata di C(7)

Link statico: $k_c = Sd(A) - Sd(C) + 1$

$= 1 + 2 + 1 = 0$

$LS(C) = A$

Chiamata di D

Link statico: $k_D = Sd(C) - Sd(D) + 1$

$= 2 - 3 + 1 = 0$

$LS(D) = C$

Chiamata di B

Link statico: $k_B = Sd(D) - Sd(B) + 1$

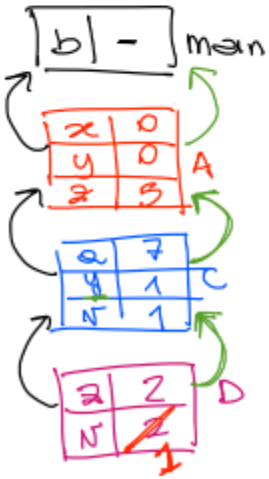
$= 3 - 2 + 1 = 2$

$LS(B) = LS(LS(D)) = A$

Esecuzione di B: $x = y + z$ (5) → $N_z = Sd(B) - Sd(A)$ = 1 → 2 ok

locale → $N_y = Sd(B) - Sd(A)$ = 1 → A $xy = 0$

Terminazione di B → Si elimina l'RdA per B

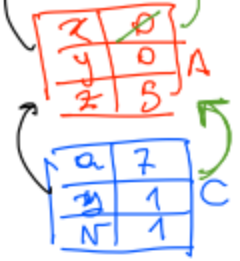


Esecuzione di D

$N = x + y$ → $N_y = Sd(D) - Sd(C) = 1 \rightarrow 1$

locale → $N_x = Sd(D) - Sd(A) = 2 \rightarrow 0$

Terminazione di D → Elimina RdA



Esecuzione di C:

$x = z + y$ → locale = 1

5 (z), 0 (y), 6 (x)

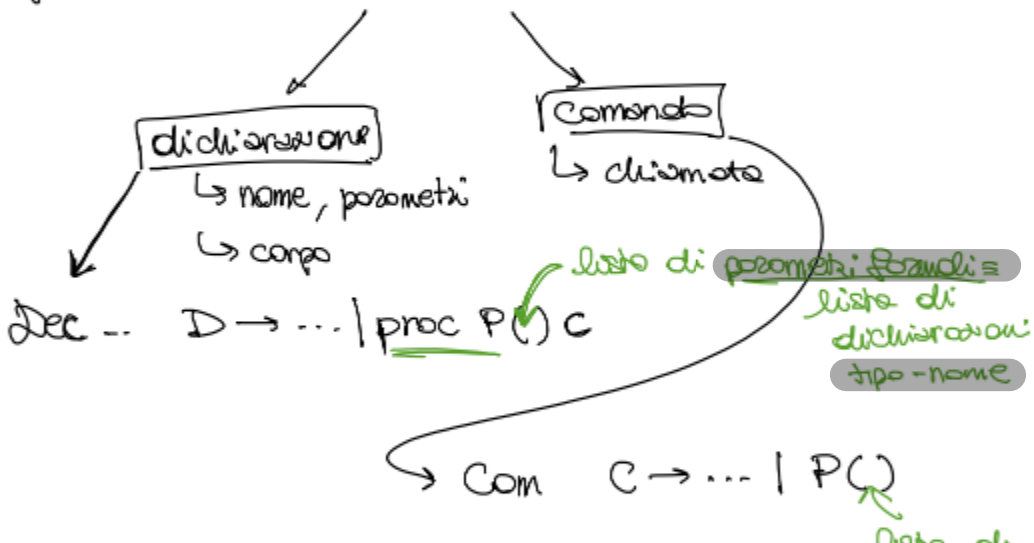
$N_z = Sd(C) - Sd(A)$

$= N_x$

$= 1$

Terminazione di C e di A e del main eliminano tutto dallo stack.

Nei linguaggi di programmazione per usare le procedure



Parametri della procedure → permettono una comunicazione tra chiamante e chiamato

Dichiarazione: `int f(int n) { return n+1; }`

- nome
- parametro formale
- tipo del valore di ritorno
- l-value (locazione)

Uso/chiamata: $x = f(3)$ $\quad x = f(x+3)$ ← r-value

Direzione della comunicazione

- In-mode
  - Chiamante ⇒ Chiamato
  - il parametro formale è ambiente locale e riceve il valore del parametro attuale
- Out-mode
  - Chiamato ⇒ Chiamante
- Inout-mode
  - Chiamato ⇔ Chiamante

Modello concettuale di passaggio
- spostiamo fisicamente il valore
- spostiamo l'accesso al valore