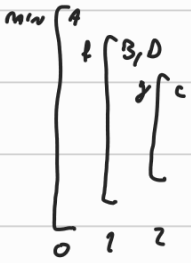
1) SCOPING STATICO / DINAMICO

- STATICO
- DINAMICO

- SUDDIVIDENDO IL PROGRAMMA SEGNANDO LE VARIABILI DICHIARATE IN QUELLA SEZIONE SEGNANDO ANCHE IL LIVELLO (SD)



SD(main) = 0

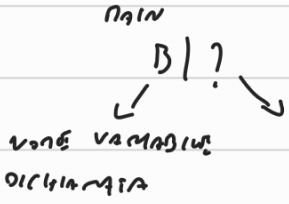
SD(f) = 1

SD(g) = 2

ORDINE DI CHIAMATA → main -> f() -> g()

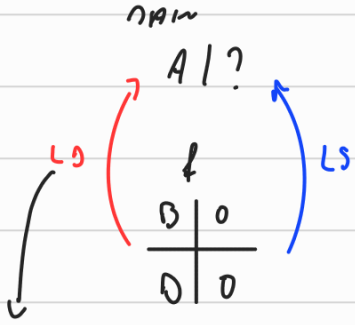
- SCOPING STATICO

PARTO DA MAIN E FACCIO



VADO ALLA FUNZIONE CHIAMATA SUBITO DOPO

main -> f



CALCOLO IL LINK STATICO

LS = SD(CHIAMANTE) − SD(CHIAMATA) + 1

LS = SD(main) − SD(f) + 1 = 0 − 1 + 1 = 0

IL LINK DINAMICO PUNTA SEMPRE ALLA FUNZIONE PRECEDENTE

N.B.

SE LS = 0 LA FRECCIA PUNTA ALLA FUNZIONE PRECEDENTE, SE LS = 1 PUNTA A 2 FUNZIONI PRIMA, SE LS = 2 PUNTA A 3 FUNZIONI PRIMA

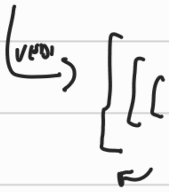
Se devo fare un operazione es. K = Z+Y (supponiamo X e Y locali)

devo trovare N_Z

↗ contenuto di LS

N_Z = SD(funzione attuale) − SD(...)

↳ cerco la prima funzione che possiede la variabile che cerco, tornando indietro di gerarchia

↳ vedi → 

a) suddivido [[[, SD, catena

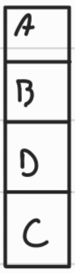
b) per ogni passaggio trovo LS

c) calcolo eventuali operazioni per ogni passaggio

Quando termino scrivo: termina tutto e lo stack viene distrutto

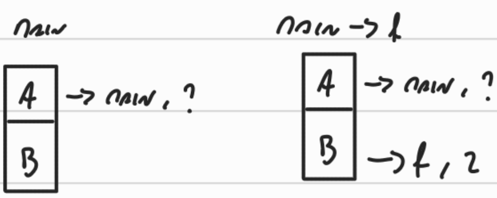
- **Scoping dinamico**

metto in colonna le variabili

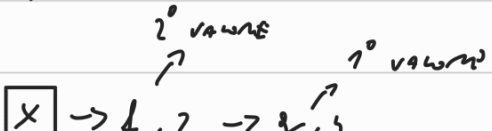


per proseguire, su ogni step (main → f, f → ..., ...), segno accanto ad ogni variabile in quale funzione e che valore assume

es:



N.B



Quindi la componente f, 2 è stata assunta da sx

Per svolgere un programma, nonostante è utilizzato come più a sx

es: x = y + z = 1 + 2

b	-> main, ?
v	-> C, 1 -> D, 7
w	-> B, 3
x	-> B, ~~3~~ 6 -> A, 6
y	-> C, 1 -> A, 0
z	-> D, 2 -> A, 5

Alla fine scrivo: termina tutto e la CRT viene distrutta

2) SCOPE STATICO / DINAMICO / SHALLOW BINDING

VARIABILI ... ①

f {

}

f1 {

③

}

VARIABILI ... ②

INT z = ...

SCOPE STATICO → USO VARIABILI AMBIENTE ①

VEDO QUINDI COSA RESTITUISCE IL RETURN DI f E f1 E COSA RESTITUISCE z

SCOPE DINAMICO

DEEP BINDING → USO VARIABILI AMBIENTE ②

VEDO QUINDI COSA RESTITUISCE IL RETURN DI f E f1 E COSA RESTITUISCE z

SHALLOW BINDING → USO VARIABILI AMBIENTE ③

VEDO QUINDI COSA RESTITUISCE IL RETURN DI f E f1 E COSA RESTITUISCE z

SE MI MANCA UNA VARIABILE (ES: y), PROVO A RICAVARLA

PS: SE INT z = $f_1(1)$ (AMBIENTE ②), POSSO PRENDERE LA VARIABILE CHE MI SERVE IN ② PERCHÉ FACCIO UN PASSO INDIETRO E VADO A VEDERE DOVE È CHIAMATA f_1

3) RICORSIONE

es. lista = (2, 4, 6)

□x → qualsiasi operazione □ con qualsiasi numero x

function ((2, 4, 6))

↳ concat (2□x, function (4, 6)) (6, 12, 18)

↳ concat (4□x, function (6)) (12, 18) → nell'es. □x = ·3

↳ concat (6□x, function ()) (18)

↳ ()

RICORSIONE IN CODA

```
LISTA FUNCTION2 (LISTA LIST, LIST RETURN) {
    IF (LENGTH(LIST) = 0) THEN RETURN RESULT;
    ELSE RETURN FUNCTION2 (CDR(LIST), CONCAT (RESULT, 3 · CAR(LIST));
}
LISTA FUN(LISTA LIST) {
    RETURN FUNCTION2(LIST, ());
}
```

3· è l'esempio, possono esserci altre operazioni

FUN ((2, 4, 6))

↳ FUNCTION2 (4, 6), CONCAT ( ) , 3 · 2) 6

↳ FUNCTION2 (6, CONCAT (6 , 3 · 4) (6, 12)

↳ FUNCTION2 ((6, 12) , 3 · 6) (6, 12, 18

4) PROGRAMMI DA COMPLETARE

AVVIANDOMI CODICE AL POSTO DI (*) E DI (**)

SCOPE STATICO

```
...:... => funzione() => ... = ...  |  f():
...:... => funzione() => ... = ...  |  i=0 => fun() => x=3
                                    |  i=2 => fun() => x=3
```

SCOPE DINAMICO

```
...:... => funzione() => ... = ...  |  f():
...:... => funzione() => ... = ...  |  i=0 => fun() => x=3
                                    |  i=2 => fun() => x=2
```

5) PASSAGGIO VALORE / RIFERIMENTO

PASSAGGIO VALORE

QUANDO PASSO LA VARIABILE NE PASSO **UNA COPIA**, IN CASO DI MODIFICARE DELLA VARIABILE, **VIENE MODIFICATA SOLO LA COPIA**

PASSAGGIO RIFERIMENTO

QUANDO PASSO LA VARIABILE NE PASSO **L'INDIRIZZO DELLA VARIABILE STESSA**, QUINDI UN RIFERIMENTO A QUELLA ORIGINALE, IN CASO DI MODIFICARE DELLA VARIABILE **VIENE MODIFICATO** IL VALORE DELLA VARIABILE ORIGINALE

6)

$\rho[\dots]$ $\quad$ es: $\rho[x \mapsto 3, z = l_x]$

$\sigma[\dots]$ memoria $\quad$ $\sigma[l_x \mapsto 2]$

$\dots := \dots$ $\quad$ $z := 5(x+z)$

devo trovare tutte le variabili necessarie e poi svolgo l'operazione

es: trovo la x

valuto y $\quad \dfrac{\rho \vdash \langle x, \sigma\rangle \to \langle 3, \sigma\rangle \quad \rho(x) = 3}{\rho \vdash \langle x+z, \sigma\rangle \to \langle 3+z, \sigma\rangle}$

valuto 5 $\quad \dfrac{\rho \vdash \langle x+z, \sigma\rangle \to \langle 3+z, \sigma\rangle}{\rho \vdash \langle 5(x+z), \sigma\rangle \to \langle 5(3+z), \sigma\rangle}$

valuto $z := \dots$ $\quad \dfrac{\rho \vdash \langle 5(x+z), \sigma\rangle \to \langle 5(3+z), \sigma\rangle}{\rho \vdash \langle z := 5(x+z), \sigma\rangle \to \langle z := 5(3+z), \sigma\rangle}$

per z è uguale, solo che la nuovo conosciamo già la x, quindi partiamo dal basso con:

$$\rho \vdash \langle z := 5(3+z), \sigma\rangle \to \langle z := 5(3+2), \sigma\rangle$$

poi scriviamo: $\rho(z) = l_z$ se ciò che cerco è in memoria

$\sigma[l_z] = 2$

poi svolgo l'operazione con:

$$\frac{\dfrac{\rho \vdash \langle 3+2, \sigma\rangle \to \langle 5(5), \sigma\rangle}{\rho \vdash \langle 5(3+2), \sigma\rangle \to \langle 5(5), \sigma\rangle}}{\rho \vdash \langle z := 5(3+z), \sigma\rangle \to \langle z := 5(5+2), \sigma\rangle}$$

$$\frac{\rho \vdash \langle 5(5), \sigma\rangle \to \langle 25, \sigma\rangle}{\rho \vdash \langle z := 5(5), \sigma\rangle \to \langle z := 25, \sigma\rangle}$$

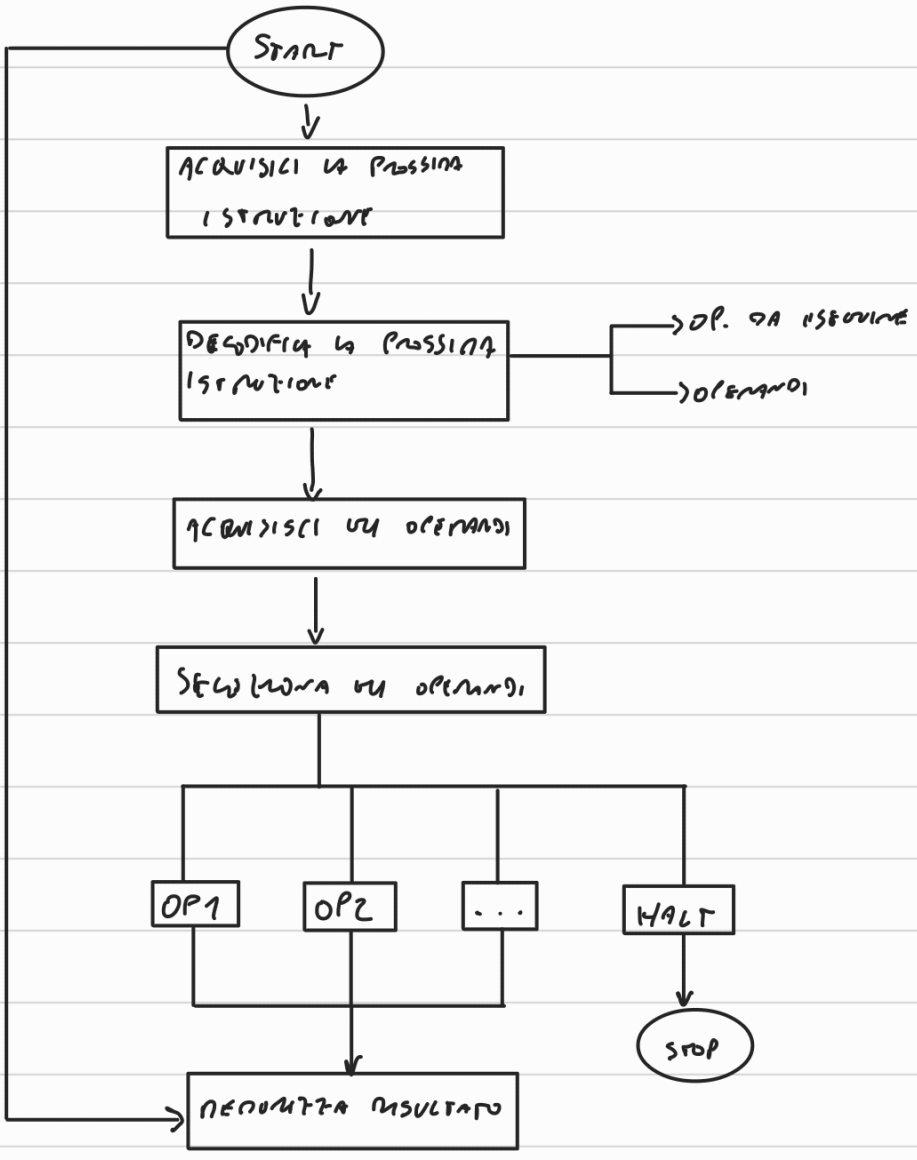
E concluso con:

$$\rho \vdash \langle z := 25, \sigma \rangle \to \rho[z] = l_z$$
$$\sigma[l_z \mapsto 25]$$

Un interprete ILL0, sviluppato in L0, è un programma che prende in input un altro programma scritto nel linguaggio L e anche un input da usare con quel programma. L'interprete esegue il programma in L passo dopo passo, utilizzando solo le funzioni e le operazioni disponibili nella macchina astratta di L0, e alla fine restituisce il risultato dell'esecuzione.

$I^{L0L} : (Prog^{L} \times Dato) \to Dato$

$[\![I^{L0L}]\!] (Prog^{L}, input) = [\![Prog^{L}]\!] (input)$



Un compilatore è un programma CLL0 che traduce (preservandone semantica e funzionalità) un programma scritto in L in un programma scritto in L0, quindi eseguibile direttamente sulla macchina astratta per L0.

$C^{LL0} : PROG^{L} \to PROG^{L0}$

$[\![C^{LL0}]\!](P^{L}) = P^{L0}$ T.C. $\forall input \in D,\ [\![P^{L}]\!](input) = [\![P^{L0}]\!](input)$