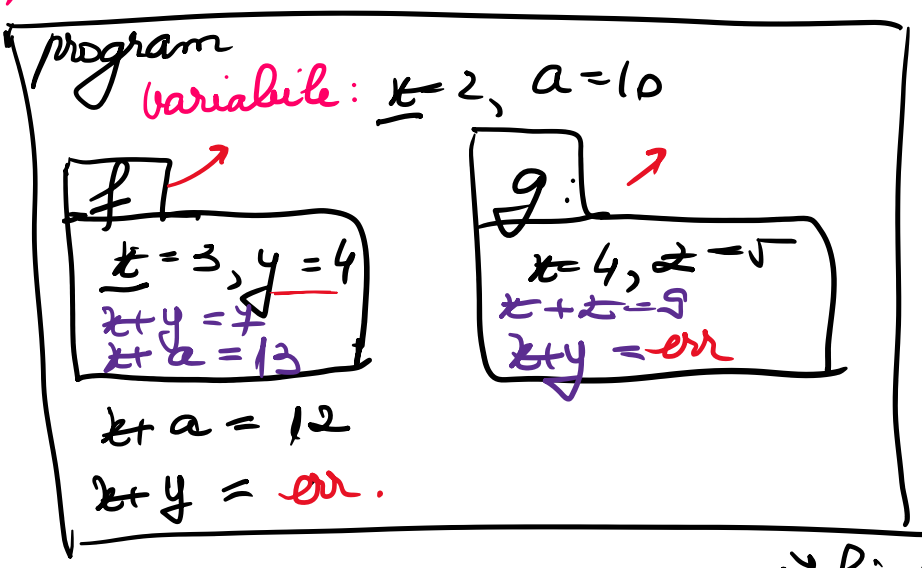
Lab 4: Legarea variabilelor + Inchideri functionale

24 March 2023 13:24

- DOMENIU DE VIZIBILITATE/SCOPE al unei var
  = zona în care variabila este vizibilă
- CONTEXT COMPUTATIONAL = mult. de (variabila, valoare)

Ex



program:
{ x=2, a=10
f, g

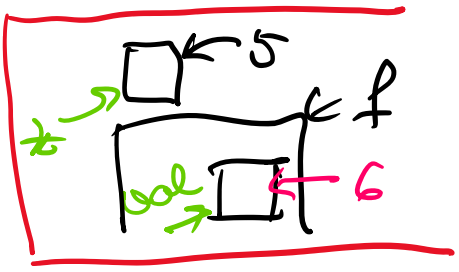
f: { x=3, y=4
a=10

g: { x=4, z=5
a=10

→ ficare var. e vizibilă în cadrul ei

LEGARE/BINDING (VS) ATRIBUIRE

```
(define x 5)
(define (f val) val)
(f 6)
```



- legare = punem un nume pe o casetă
- atribuire = punem o val. în casetă

! În Racket nu putem lega un nume de 2 ori definite în același scope.

```
(define n 10)
...
(define n 5)  X
```

(define n 5)

LEGĂRI ÎN RACKET

① DEFINE → legare globală

- (define x 7) → x se leagă la 5, și e vizibil global

- (define (f x)
  (define (g y) (+ y 1))
  (+ x (g x)))

② când apelăm funcții

(define (f x) (+ x 7))

(f 2) → x (param. formal) se leagă la 2 (param. actual)

③ LET → introduce un nou scope

- SINTAXĂ

(let ([...] ... [...]) → definiții: ([x 1] [y 2])
  expr 1
  ...
  expr n)
→ valoarea de return

Ex:

(let ([x 2] [y 3])
  (let ([x 4]

(list x y)
(cons x y))

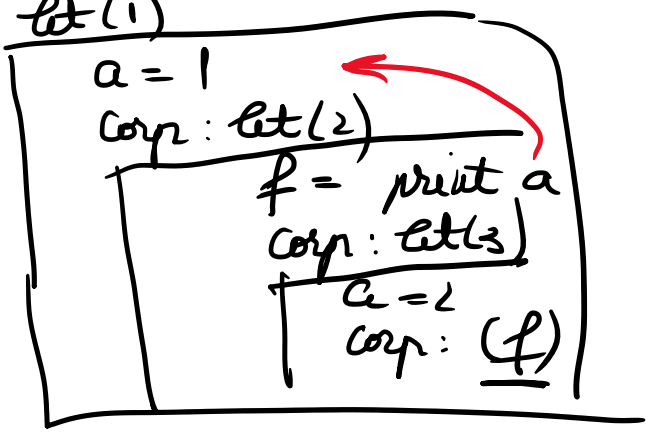
→ (2 . 3)

- VIZIBILITATE : doar în corp

LEGARE STATICĂ ⓦ DINAMICĂ

- STATICĂ (în Racket)

```
(let ([a 1])
  (let ([f (λ () (print a))])
    (let ([a 2])
      (f))))
(display "\n")
```



f - se leagă la (print a)
a - cel mai aproape : 1

⇒ 1

- DINAMICĂ : 2 (cea mai apropiată legare a lui a este 2)

(4) let* → la fel ca let, dar:
VIZIBILITATE → următoarele definiții
→ Corpul let*

(5) letrec
VIZIBILITATE → toată zona de def.
→ Corpul letrec

```
(letrec ([a (+ b 1)] [b 2])
  (cons a b))
```

(⊃ ().
(define a NULL)
(define b NULL)
(a → b+1)

(a → b+1)
(b → 2)

⑥ NAMED LET

→ pt. a construi loops

→ pt. a defini fct. helper locale intr-o funcție.

ÎNCHIDERI FUNCȚIONALE

= pereche [text, context]

Ex:

```
(define n 10)
(define f (λ (x) (+ x n)))
```

(n 10)

text