Paradigme de Programare

Conf. dr. ing. Andrei Olaru

andrei.olaru@cs.pub.ro | cs@andreiolaru.ro
Departamentul de Calculatoare

2018

Cursul 5

Anexă: Legare dinamică și efecte laterale în Racket

Cursul 5: Anexă: Legare dinamică și efecte laterale în Racket

Anexă: Evaluare și legarea variabilelor în Racket

Legare dinamică și închideri funcţionale
Setup

- Racket nu suportă re-definirea unui simbol prin `define`.
- Pentru rularea acestor exemple, alegeți limbajul Pretty Big pentru execuție.

Legare dinamică și închideri funcţionale
Exemplul 1

Ex Exemplu

```
1 (define x 0)
2 (define f (lambda () x))
3 (f)
4 (define x 1)
5 (f)
```

Output:

Legare dinamică și închideri funcţionale
Exemplul 1

Ex Exemplu

```
1 (define x 0)
2 (define f (lambda () x))
3 (f)
4 (define x 1)
5 (f)
```

Output: `0 1`

Legare dinamică și închideri funcţionale
Exemplul 2 (mai obscur)

Ex Exemplu

```
1  (define factorial (lambda (n)
2      (if (zero? n) 1
3          (* n (factorial (- n 1))))))
4
5  (factorial 5)
6
7  (define g factorial)
8  (define factorial (lambda (x) x))
9
10 (g 5)
```

Output:

Legare dinamică și închideri funcţionale

Exemplul 2 (mai obscur)

Exemplu

```
1  (define factorial (lambda (n)
2      (if (zero? n) 1
3          (* n (factorial (- n 1))))))
4
5  (factorial 5)
6
7  (define g factorial)
8  (define factorial (lambda (x) x))
9
10 (g 5)
```

Output: 120 20

Legare dinamică și închideri funcţionale
Exemplul 3

Exemplu

```
1  (define x 0)
2  (define f (lambda () x))
3  (define x 1)
4
5  (define g
6    (lambda (x)
7      (f)))
8
9  (g 2)
```

Output:

Legare dinamică și închideri funcţionale
Exemplul 3

Exemplu

```
1  (define x 0)
2  (define f (lambda () x))
3  (define x 1)
4
5  (define g
6    (lambda (x)
7      (f)))
8
9  (g 2)
```

Output: `1`

Legare dinamică și închideri funcţionale
Exemplul 4

Exemplu

```
1  (define comp
2    (lambda (f) (lambda (g) (lambda (x) (f (g x))))))
3  (define inc (lambda (x) (+ x 1)))
4  (define comp-inc (comp inc))
5
6  (define double (lambda (x) (* x 2)))
7  (define comp-inc-double (comp-inc double))
8  (comp-inc-double 5) ; 11
9
10 (define inc (lambda (x) x))
11 (comp-inc-double 5) ; tot 11
```

Legare dinamică și închideri funcţionale

Exemplul 4 – comentarii

- $comp\text{-}inc \equiv < \lambda g.\lambda x.(f\ (g\ x)); \{f \leftarrow \lambda x.(+\ x\ 1)\} >$
- $comp\text{-}inc\text{-}double \equiv$
  $< \lambda x.(f\ (g\ x)); \{f \leftarrow \lambda x.(+\ x\ 1), g \leftarrow \lambda x.(*\ x\ 2)\} >$
- Inutilitatea redefinirii lui `inc`: valoarea sa (o funcție) fusese deja salvată în context, în momentul aplicării

Anexă: Efecte laterale

Construcţia `set!`

- Modifică valoarea unei variabile

  Ex | Exemplu

```
1  (define x 0)
2  (define f (lambda (p)
3       (set! x p)
4       x))
5  (f 3) ; 3
6  x ; 3
```

- Diferenţă la nivel de intenţie faţă de `let`-uri şi `define`, care urmăresc definirea de variabile noi şi nu modificarea celor existente!

Atribuiri

- Avantaje:
  - Modelarea obiectelor a căror stare variază în timp
  - Evitarea pasării explicite a fiecărei modificări de stare

- Dezavantaj: pierderea transparenței referențiale.