Programare funcţională – Laboratorul 8

Chirurgie, Expresii lambda, Liste circulare, Mapare

Isabela Drămnesc

April 17, 2012

1 Concepte

- append, nconc
- reverse, nreverse
- remove, delete
- remove-if, delete-if
- expresii lambda
- apply, funcall
- mapcar, maplist

2 Întrebări din Laboratorul 6

- Ce alte forme ale lui DO mai cunoaştem? Daţi câte un exemplu pentru funcţionalitatea fiecărei forme!
- Care e diferenţa între append şi nconc? Reprezentaţi folosind celule de reprezentare comportamentul lui nconc. Definiţi o funcţie proprie care să simuleze comportamentul lui nconc.
- Care e diferenţa dintre reverse şi nreverse? Reprezentaţi folosind celule de reprezentare comportamentul lui nreverse. Definiţi o funcţie proprie care să simuleze comportamentul lui nreverse.
- Scrieţi (în cel puţin două moduri) câte o definiţie iterativă în Lisp pentru:
  1. cmmdc(a,b)
  2. factorial(n)
  3. inversa(lista)
  4. lungime(lista)

3 Exerciţii

3.1 remove, delete

```
> (setq l '(azi e soare))

> (remove 'soare l)

> l

> (delete 'soare l)

> l
```

Observaţie:

La utilizarea funcţiilor chirurgicale este important să ştim că, deşi rezultatul întors e corect, efectul lateral nu apare în următoarele cazuri:

- când utilizăm nconc şi prima listă este nil;
- când utilizăm delete şi ştergem primul element din listă;

Exemple:

```
> (setq *l* nil)

> *l*

> (nconc *l* '(a b c))

> *l*

> (setq *ll* '(a b c d))

> (delete 'a *ll*)

> *ll*

> (delete 'b *ll*)

> *ll*
```

3.2 remove-if, delete-if, subst

```
> (subst 'a 'b '(a b (b (d e f) b) b))

> (setq w '(1 + 1 + 1 = 3))

> w

> (remove-if 'numberp w)
```

```
> w

> (delete-if 'numberp w)

> w
```

3.3 Expresii lambda

Apar:

- când o funcţie e folosită o singură dată şi e mult prea simplă ca să merite a fi definită;
- funcţia de aplicat trebuie sintetizată dinamic (fiind imposibil să fie definită cu DEFUN).

Sintaxă:
(lambda l f1 f2 f3 ... fn)
sau
((lambda l f1 f2 f3 ... fn) par1 par2 ... parn)

- defineşte o funcţie utilizată local;
- l reprezintă lista parametrilor; (poate fi dată explicit (număr fix de parametri) sau parametric (lista l având un număr variabil de parametri);
- f1 f2 ... fn corpul funţiei;

Argumentele sunt evaluate la apel. Dacă se doreşte ca argumentele să nu fie evaluate la apel trebuie folosită forma QLAMBDA.
Exemple:

```
> ((lambda () 20))

> ((lambda (x) (1+ x)) 5)

> ((lambda (x y) (+ x y)) 5 7)

> ((lambda (y)
   ((lambda (x) (+ x (* 2 y)
                     (* 12 x y)
                     (* (* x x) (* y y))
                  )
    )
    21
   )
  )
  2
  )

> ((lambda (x y)
      (+ x
            (* 2 y)
            (* 12 x y)
```

```
                (* (* x x) (* y y))))
 21 2)
```

Expresiile lambda pot fi considerate funcţii anonime (fără nume).
Exemplu:

```
(setf (symbol-function 'aduna_cu_3)
      '(lambda (x) (+ x 3)))

>(aduna_cu_3 26.6)
```

3.4 apply, funcall

Există cazuri în care numărul parametrilor unei funcţii trebuie stabilit dinamic. Aplicarea unei funcţii asupra unei mulţimi de parametri sintetizată eventual dinamic este posibilă cu ajutorul funcţiilor APPLY şi FUNCALL.
Sintaxă:
(apply functie (listaparametri))

Exemple:

```
> (apply 'cons '(a b))

> (apply 'max '(1 2 3 4 5 6))

> (apply '+ '(1 2 3 4))
```

Sintaxă:
(funcall functie arg1 arg2 ...)

Este o variantă a funcţiei apply care permite aplicarea unei funcţii la un număr fix de parametri.
Exemple:

```
> (funcall 'cons 'a 'b)

> (funcall 'max 1 2 3 4 5 6)

> (setq p 'car)

> ((eval p) '(a b c))
error: bad function - (EVAL P)   ; incorect
     ;devine corect daca se utilizeaza apply sau funcall

> (setq p 'car)

> p

> (funcall p '(a b c))

> (apply p '((a b c)))
```

```
> (setq f '(lambda(x) (+ x 3)))

> f

> (f 6)

> (funcall f 6)
```

3.5 Modificarea dinamica a funcţiilor

Deoarece funcţiile sunt reprezentate prin liste, ca nişte date, ele pot fi modificate dinamic ca orice dată în timpul programului.
Exemplu:

```
> (setq f '(lambda(x) (+ x 3)))

> (funcall f 5)

> (setf (third (third f)) 4)

> f

> (funcall f 5)
```

3.6 mapcar, maplist

Mapcar este o funcţie de aplicare globală. Ea se aplică pe rând asupra elementelor listei argument, rezultatele fiind culese într-o listă întoarsă ca valoare a apelului funcţiei MAPCAR. Evaluarea se încheie la terminarea listei celei mai scurte.

```
> (mapcar 'oddp '(1 2 3))
(T NIL T)

> (mapcar 'list '(1 2 3))
((1) (2) (3))

> (mapcar '+ '(1 2 3) '(4 5 6))
(5 7 9)

> (mapcar 'cons '(1 2 3) '(4 5 6))
((1 . 4) (2 . 5) (3 . 6))

> (maplist 'length '(1 2 3))
(3 2 1)

> (mapcar 'list '(a b c) '(1 2))
((A 1) (B 2))

> (mapcar 'car '((a b c) (x y z)))
```

```
(A X)
```

MAPLIST se aplică tuturor listelor, apoi cdr-urilor listelor, apoi cddr-urilor acestora (tuturor sublistelor succesive) până când una din liste ajunge la nil. Se returnează lista rezultatelor succesive obţinute.

```
> (maplist 'append '(a b c) '(1 2 3))
((A B C 1 2 3) (B C 2 3) (C 3))

> (maplist '(lambda (x) x) '(a b c))
((A B C) (B C) (C))

> (maplist 'list '(a b c))
(((A B C)) ((B C)) ((C)))

> (maplist 'length '(1 2 3))
(3 2 1)
```

3.7 Liste circulare

Studiaţi următorul exemplu:

```
(defun list-len (x)
  (do ((n 0 (+ n 2))                ;Contor
       (fast x (cddr fast))         ;Pointer rapid: merge din 2 in 2
       (slow x (cdr slow)))         ;Pointer incet: trece prin fiecare cdr
       (nil)
       ;; Daca pointerul rapid atinge sfarsitul intoarce n.
       (when (endp fast) (return n))
       ;; Daca cdr-ul pointerului rapid atinge este cel final, intoarce n+1.
       (when (endp (cdr fast)) (return (+ n 1)))
       ;; Daca pointerul rapid il ajunge din urma pe cel incet,
       ;; inseamna ca avem de-a face cu o lista circulara.
       ;; Returnam nil.
       (when (and (eq fast slow) (> n 0)) (return nil))))
```

Testaţi de asemenea următorul exemplu:

```
(setq x '(1 2))

> (rplacd x x)

> (list-len x)
```

3.8 Propriul nostru predicat de egalitate

Functie ce decide "egalitatea" a doua structuri formate cu cons-uri:

```
(defun egal (l1 l2)
  (cond ((and (null l1) (null l2)) t)
        ((and (atom l1) (atom l2)) (equal l1 l2))
```

```
        ((and (atom l1) (not (atom l2))) nil)
        ((and (not (atom l1)) (atom l2)) nil)
        ((egal (car l1) (car l2)) (egal (cdr l1) (cdr l2)))
        (t nil)
  )
)
```

4 Tema

1. Evaluati următoarele s-expresii:

```
> (cadr '(a b c d e))

> (second '(a b c d e))

> (nth 2 '(a b c d e))

> (cadr (cadr '((a b c) (d e f) (g h i))))

> (cadadr '((a b c) (d e f) (g h i)))

> (cddadr '((a b c) (d e f) (g h i)))

> (last '((a b c) (d e f) (g h i)))

> (third '((a b c) (d e f) (g h i)))

> (cdr (third '((a b c) (d e f) (g h i))))
```

2. Definiţi o funcţie iterativă RANGE care primeşte ca şi parametru o listă de numere şi returnează o listă de lungime 2 care conţine cel mai mic număr şi cel mai mare număr. Utilizaţi predicate > şi < Asiguraţi-vă că lista e parcursă o dată. Scrieţi încă o funcţie VALID-RANGE, care returnează acelaşi rezultat ca RANGE dacă elementele listei sunt toate numere şi dacă nu returnează INVALID.
Exemplu:

```
> (range '(0 7 8 2 3 -1))
(-1 8)
> (range '(7 6 5 4 3))
(3 7)
> (valid-range '('a 7 8 2 3 -1))
INVALID
> (valid-range '(0 7 8 2 3 -1))
(-1 8)
```

3. Scrieţi 2 s-expresii pentru a accesa simbolul C pentru fiecare din listele următoare:

(a) (A B C D E)

(b) ((A B C) (D E F))

(c) ((A B) (C D) (E F))

4. Cum schimbaţi C în SEE pentru fiecare din listele următoare (fără a utiliza subst):

(a) (A B C D E)

(b) ((A B C) (D E F))

(c) ((A B) (C D) (E F))

(d) (A (B C D) E F)

Notă: Termen de realizare: laboratorul următor.