Programare funcţională – Laboratorul 4

Recursivitate, Recursivitate finală

Isabela Drămnesc

March 20, 2012

1 Concepte

- Recursivitate
- Recursivitate finala
- Tehnica variabilei colectoare
- Definire de funcţii final recursive
- Operaţii asupra listelor la nivel superficial
- Operaţii asupra listelor la orice nivel
- Operaţii cu mulţimi
- Utilizare (time <expresie>)
- Utilizare (trace <expresie>)

2 Întrebări din Laboratorul 3

- În ce situaţii apar variabilele în Lisp? Cum se numesc în fiecare caz. Daţi exemple pentru a evidenţia diferenţa dintre ele.
- Ce este recursivitatea? Daţi cel puţin 2 exemple de funcţii recursive (scrieţi definiţia).
- Cât de importantă este ordinea declarării clauzelor în definirea unei funcţii recursive în Lisp?
- Scrieţi o funcţie în Lisp pentru:
  1. Concatenarea a doua liste;
  2. Inversa unei liste;
  3. Lungimea unei liste.

3 Recursivitate finală. Metoda variabilei colectoare

O funcţie este *liniar recursivă* dacă se apelează o singură dată pe ea insăşi pentru a întoarce rezultatul.

O funcţie este *"gras"/"exploziv" recursivă* dacă se apelează de mai multe ori pe ea insăşi pentru a întoarce rezultatul.

O funcţie recursivă este *final-recursivă* dacă:

- apelurile recursive nu sunt argumente pentru alte funcţii şi nu sunt utilizate ca şi teste;
- dacă valoarea obţinută pe ultimul nivel de recursivitate rămâne neschimbată până la revenirea pe nivelul de sus;
- la ultima copie creată se obţine rezulatul, rezultat ce rămâne neschimbat la revenire;

Exemplele cu definirea de funcţii recursive de până acum reprezintă funcţii nefinal-recursive. La astfel de funcţii nefinal recursive apelul recursiv este conţinut într-un apel de funcţie (+, -, cons, append etc).

Recursivitatea grasă este foarte ineficientă, recursivitatea finală este cea mai eficientă.

Pentru transformarea unei funcţii recursive într-o funcţie final-recursivă se foloseşte tehnica variabilelor colectoare.

3.1 Factorial

```
;testare functii definite in lab3

;>(time (fact 2485))

;>(time (factorial-cond 2484))

;>(time (factorialn 2000))

;; tehnica variabilei colectoare ;;

(defun factf (n)
       (fact-aux n 1))

(defun fact-aux (n rez)
       (cond ((= n 0) rez)
             (t (fact-aux (- n 1) (* n rez)))
       )
)
```

```
;>(time (factf 2700))

;>(trace fact-aux)

;>(fact-aux 5 1)
```

3.2 Fibonacci

```
(defun fib (n)
  (if (< n 2) n
    (+ (fib (- n 2)) (fib (- n 1)))
  )
)

;>(time (fib 3000))

;;;; Varianta final recursiva

(defun rfib (n)
  (if (< n 2) n
    (fib-aux n 1 0)
  )
)

(defun fib-aux (n fn fn-1)
  (if (= 1 n) fn
    (fib-aux (- n 1) (+ fn fn-1) fn)
  )
)

; >(time (rfib 3000))

;> (trace fib-aux)
```

3.3 Inversa

```
;;; definire reverse cu o variabila colectoare ;;
(defun rev1 (l1 l2)
       "are 2 argumente reverse-ul definit de noi"
       (cond ((null l1) l2)
             (t (rev1 (cdr l1) (cons (car l1) l2) ))
       )
)

(defun rev-list (lista)
       (rev1 lista ())
)

#| > (rev-list '(1 2 3 4 (8 9)))
```

```
((8 9) 4 3 2 1)

> (rev1 '(1 2 3) '())
(3 2 1)

> (trace rev1)

>(rev1 '(1 2 3) '())
|#
```

3.4 Lungimea unei liste

Urmând exemplele de mai sus scrieţi o funcţie final recursivă care returnează lungimea unei liste. De exemplu:

```
> (rlen '(1 2 3 4))
4

> (rlen '(1 2 3 (j k) (m n o p) 4))
6
```

4 Nivel superficial, Orice nivel

4.1 Definiţi o funcţie în Lisp care determină primul atom dintr-o listă. Funcţia se va comporta astfel:

La nivel superficial:

```
>(prim-elem '(1 2 3))
1

>(prim-elem '())
NIL

>(prim-elem '((a b) (c d e) l (o p)))
L
```

La orice nivel:

```
>(prim-2 '((((2 3 4) 8) 8 9) 9))
2

>(prim-2 '((((a b 4) 8) 8 9) 9))
A
```

5 Tema

5.1 Ridicare la putere - final recursivă

Scrieţi o funcţie final recursivă pentru calculul x^y.

5.2 Adună numere din listă - final recursivă

Scrieţi o funcţie final recursivă care returnează suma numerelor dintr-o listă. Peste elementele listei care sunt simboluri sare (nu le adună), adună doar numerele din listă. Exemplu:

```
>(aduna-numere '(1 2 d 4))
7
>(aduna-numere '(1 2 d 4 (5 lalala 5)))
17
```

5.3 Mulţimi - operaţii

Fiind date două mulţimi A şi B, să se scrie o funcţie recursivă ce determină reuniunea celor două mulţimi ($A \cup B$). Similar, să se scrie câte o funcţie pentru calculul intersecţiei ($A \cap B$), diferenţei ($A\ B$) şi diferenţei simetrice ($A \triangle B$).

Notă: Termen de realizare: laboratorul următor.