

Programare funcțională – Laboratorul 4

Recursivitate, Recursivitate finală

Isabela Drămnesc

1 Concepte

- Recursivitate
- Recursivitate finală
- Tehnica variabilei colectoare
- Definire de funcții final recursive
- Operații asupra listelor la nivel superficial
- Operații asupra listelor la orice nivel
- Operații cu mulțimi
- Utilizare (time <expresie>)
- Utilizare (trace <expresie>)

2 Întrebări din Laboratorul anterior

- În ce situații apar variabilele în Lisp? Cum se numesc în fiecare caz. Dați exemple pentru a evidenția diferența dintre ele.
- Ce este recursivitatea? Dați cel puțin 2 exemple de funcții recursive (scrieți definiția).
- Cât de importantă este ordinea declarării clauzelor în definirea unei funcții recursive în Lisp?
- Scrieți o funcție în Lisp pentru:
 1. Concatenarea a doua liste;
 2. Inversa unei liste;
 3. Lungimea unei liste.

3 Recursivitate finală. Metoda variabilei colectoare

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

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

O funcție recursivă este *final-recursivă* dacă:

- apelurile recursive nu sunt argumente pentru alte funcții și nu sunt utilizate ca și teste;
- dacă valoarea obținută pe ultimul nivel de recursivitate rămâne neschimbată până la revenirea pe nivelul de sus;
- la ultima copie creată se obține rezultatul, rezultat ce rămâne neschimbat la revenire;

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

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

Pentru transformarea unei funcții recursive într-o funcție final-recursivă se folosește 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 scrieți o funcție 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 Definiți o funcție în Lisp care determină primul atom dintr-o listă. Funcția 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

Scrieți o funcție final recursivă pentru calculul x^y .

5.2 Adună numere din listă - final recursivă

Scrieți o funcție 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 Mulțimi - operații

Fiind date două mulțimi A și B, să se scrie o funcție recursivă ce determină reuniunea celor două mulțimi ($A \cup B$). Similar, să se scrie câte o funcție pentru calculul intersecției ($A \cap B$), diferenței ($A \setminus B$) și diferenței simetrice ($A \Delta B$).

Notă: Termen de realizare: laboratorul următor.