Programare funcţională – Laboratorul 3

Definire de noi funcţii, Recursivitate

Isabela Drămnesc

March 14, 2012

1 Concepte

- Variabile legate şi libere
- Funcţii recursive
- Recursivitate simpla şi dublă
- Final recursivitate
- Recursivitate compusă
- Recursivitate monotonă şi nemonotonă
- Operaţii cu liste, mulţimi, vectori şi matrici

2 Întrebări din laboratorul 2

- Care este diferenţa între SET, SETQ, SETF?
- Ilustraţi prin cel puţin un elemplu aceasta!
- Cum definim EQ, EQL, EQUAL?
- Ilustraţi diferenţa dintre ele prin cel puţin un exemplu;
- De ce funcţia PRINT afişează de două ori pe ecran?
- Cum folosim IF? (Sintaxa, exemplu)
- Cum folosim COND? (Sintaxa, exemplu)

3 Completări

Tipul complex

Numerele complexe sunt notate ca o secvenţă:

```
#c(<real>, <imaginar>), unde
   <real> si <imaginar> sunt numere in acelasi format.
```

Exemple de numere complexe:

```
#c(2 3)                              ; numarul 2 + 3*i
#c(0 1)                              ; numarul i
#c(2/3 1.3)                 ; convertit intern la #c(0.6666667 1.3)
```

Aflarea tipului

```
> (type-of 23.3)
SINGLE-FLOAT

> (type-of 23.)
(INTEGER 0 16777215)

> (type-of 23)
(INTEGER 0 16777215)

> (type-of -23)
(INTEGER -16777216 (0))

> (type-of 'lala)
SYMBOL

> (type-of '(1 2 3))
CONS

> (type-of #c(3 4))
COMPLEX
```

4 Variabile legate şi variabile libere

Variabilele în Lisp se găsesc într-una din situaţiile:

- ca argumente în *set*, *setq*, *pset*, *psetq*, *setf*, sau *defvar*; În acest caz se mai numesc şi *variabile globale*;
- ca variabilă în lista de parametri ai unei definiţii de funcţie; În acest caz se numesc *variabile locale*;
  - dacă variabila apare în lista de parametri ai unei funcţii ea se mai numeşte *variabila legata* în raport cu acea funcţie;
  - dacă variabila apare în corpul unei funcţii şi nu apare în lista de parametri se mai numeşte *variabila libera* în raport cu acea funcţie.

Exemple:

Analizaţi următoarele exemple şi trageţi concluziile:
Exemplu 1)

```
> (setq x 10 y 20)
20
```

```
> (defun f1 (x)
      (+ x y))
F1
> (f1 3)
23
> x
10
> y
20
```

x e variabilă legată;
y e variabilă liberă;

Exemplu 2)

```
> (setq x 100 y 200)
200
> x
100
> y
200
> (defun f2 (x)
     (setq x 10)
     (+ x y))
F2
> (f2 x)
210
> (f2 8)
210
> x
100
> y
200
```

x este variabila globală şi variabilă locală;
valoarea lui x a fost schimbată în interiorul funcţiei $f2$, dar la ieşirea din funcţie x va avea valoarea de dinainte;

Exemplu 3)

```
> (setq x 10 y 20)
20
> (defun f3 (x)
          (setq x 100 y 200)
          (+ x y))
F3
> (f3 x)
300
> (f3 400)
300
> (f3 lala)
error: unbound variable - LALA
```

```
> (f3 19292.34)
300
> x
10
> y
200                ;Explicati !!!
```

Funcţiile în Lisp pot fi imbricate, o funcţie poate fi apelată din altă funcţie.

5 Probleme

5.1 (Creare-Scriere-Compilare-Utilizare fişier Lisp)

Creaţi un fişier lab3.lsp Acest fişier va conţine definiţii de funcţii:

1) O funcţie care primeşte ca parametru o listă cu două elemente şi returnează lista cu elementele inversate;

```
;;; functie pentru printarea unei liste cu doua elemente
(defun print-list-2 (el1 el2)
       "Printeaza o lista formata din cele doua elemente"
       (list el1 el2)
)

(defun inversare (lista)
       "Inverseaza elementele unei liste cu 2 elemente"
       (print-list-2 (cadr lista) (car lista))
)

#| >(inversare '(2 3))
   (3 2)

   >(documentation 'print-list-2 'function)
    "Printeaza o lista formata din cele doua elemente"   |#

; sau direct
(defun inversare-2 (lista)
       (list (cadr lista) (car lista))
)
```

2) O funcţie care returnează mediana a trei elemente, funcţia primeşte trei argumente numerice şi returnează pe cel din mijloc ca valoare (adică pe cel care nu e cel mai mic şi nu e cel mai mare).
Compilaţi fişierul utilizând:

```
> (compile-file "lab3.lsp")

> (load "lab3")
```

Apoi verificaţi (încercând mai multe exemple) dacă aţi definit corect funcţiile.

5.2 (Ecuaţia de gradul al doilea)

Scrieţi un program Lisp pentru calcularea soluţiilor ecuaţiei de gradul al doilea ştiind că:

Ecuaţia are forma $a \cdot x^2 + b \cdot x + c = 0$
Dacă $a = 0$ atunci soluţia ecuaţiei este $\frac{-c}{b}$
Delta este $\Delta = b^2 - 4 \cdot a \cdot c$

- Dacă $\Delta < 0$, atunci ecuaţia nu are soluţii reale;
- Dacă $\Delta = 0$, atunci soluţiile ecuaţiei vor fi $x_1 = x_2 = \frac{-b}{2 \cdot a}$
- Dacă $\Delta > 0$, atunci soluţiile ecuaţiei vor fi $x_1 = \frac{-b - \sqrt{\Delta}}{2 \cdot a}$ şi $x_2 = \frac{-b + \sqrt{\Delta}}{2 \cdot a}$

6 Recursivitate

Un obiect este recursiv dacă este definit în funcţie de el însăşi.

Ideea este asemănătoare cu demonstrarea unei proprietăţi prin inducţie matematică (inducţia structurală):

- avem cazul (cazurile) de bază (cel(e) mai simplu (simple));
- relaţia care exprimă cazul general în funcţie de cazuri mai simple.

$$f[x] = \begin{cases} \text{valoarea pentru cazul cel mai simplu;} \\ \\ \text{expresie de reducere a cazului general la un caz mai simplu.} \end{cases}$$

Câteva exemple de funcţii recursive se găsesc în laboratorul 1.
Exemple:

6.1 Factorial

$$n! = f[n] = \begin{cases} 1 & \text{daca } n = 1; \\ f[n-1] * n & \text{altfel.} \end{cases}$$

în Lisp:
Varianta 1)

```
(defun fact (n)
        (if (zerop n)       ;conditia de oprire
                    1       ; 1 <-- (fact 0)
                  (* n (fact (- n 1)))     ;apelul recursiv
         )
)

>(trace fact)

>(fact 4)

>(untrace fact)
```

Varianta 2)

```
(defun factorial-cond (n)
        (cond ((= n 0) 1)
              (t (* n (factorial-cond (- n 1))))
        )
)
```

Varianta 3) !!!! ATENTIE foarte mare la ordinea condiţiilor
Prima dată scriem condiţia pentru obiectul cel mai simplu;
Apoi scriem apelul recursiv!
Pentru următorul exemplu vom obţine "stack overflow"

```
(defun factorialn (n)
        (cond
              (t (* n (factorialn (- n 1))))
                    ((= n 0) 1)
        )
)
```

6.2 Fibonacci (Recursivitate dublă)

Funcţia care calculează termenul n din şirul Fibonacci: 1,1,2,3,5,8,13,21,34,55,89,144,...

Ştim că:

$F_0 = 1$

$F_1 = 1$

$F_n = F_{n-1} + F_{n-2}$ pentru $n > 1$.

Varianta Lisp:

```
(defun fibonacci (n)
  (cond ((= n 0) 0)
        ((= n 1) 1)
        (t (+ (fibonacci (- n 1))
              (fibonacci (- n 2))
           )
        )
  )
)
```

6.3 X la puterea Y

Problema 7.2.1 (din laboratorul 2)

```
;;; 1) nu se analizeaza cazul cand y e negativ
(defun puterexy (x y)
       (cond ((zerop y) 1)
             (t (* x (puterexy x (- y 1))))
       )
)
```

```
;;; 2) aceeasi functie folosind cond
;returneaza valoarea cand exponentul e negativ
(defun putere2-x-y (x y)
      (cond ((= y 0) "exponentul e 0, deci rezultatul e " 1)
            ((> y 0) (*  x (putere2-x-y x (- y 1))))
            (t "y e negativ, iar rezultatul este "
                        (/ 1 (putere2-x-y x (- y))))
      )
)
;interogari
> (putere2-x-y 2 40)
1099511627776

> (putere2-x-y 2 -40)
1/1099511627776

> (putere2-x-y 10 0)
1

> (putere2-x-y 10000 0)
1

#| 3) Conteaza ordinea clauzelor in cond in
cazul acesta? Dar in general?    |#
(defun expo (x y)
    "Functia care calculeaza x^y"
       (cond ((< y 0) "exponent negativ !!" (/ 1 (expo x (- y))))
             ((= y 0) "orice numar la puterea 0 este " 1)
             (t (* x (expo x (- y 1))))
       )
)

;; interogari
>(documentation 'expo 'function)
"Functia care calculeaza x^y"

> (expo 2 70)
error: argument stack overflow

> (expo 2 60)
1152921504606846976

> (expo 2 69)
590295810358705651712

> (expo 2 69.6)
error: argument stack overflow

;;;; 4) folosim if in if (in loc de cond)
```

```
(defun expo2 (x y)
        (if (> y 0)
             (* x (expo x (- y 1)))
             (if (= y 0) 1
                         (/ 1 (expo2 x (- y))))
        )
)

;;interogari
> (expo2 2 0)
1

> (expo2 2 -9)
1/512

> (expo2 2 9)
512
```

6.4 Scrieţi o funcţie pentru calculul sumei elementelor unei liste

6.5 Discutarea Problemei 7.2.3 (funcţie care returnează numărul de numere care apar într-o listă) din laboratorul 2.

7 Temă

7.1 CMMDC-iterativ şi recursiv

Scrieţi o funcţie care calculează cmmdc a două numere.
1) Se foloseşte definiţia lui Euclid prin scăderi repetate.
2) Se foloseşte definiţia recursivă a lui Euclid: Fie a şi b două numere întregi pozitive. Dacă b=0, atunci cmmdc(a,b)=a; altfel cmmdc(a,b)=cmmdc(b,r), unde r este restul împărţirii lui a la b.

7.2 Calcularea mediei aritmetice a elementelor unei liste.

7.3 Recunoaşterea palindroamelor.

Palindrom[1]. [sursa:wikipedia.org]

7.4 !!! Suplimentar !!!

Obligatoriu în timp (Probleme)

[1]Un palindrom este un cuvânt, frază, număr (sau orice altă secvenţă de obiecte) care are proprietatea că citită/ (parcursă) din orice direcţie arată la fel (ajustarea punctuaţiei şi spaţiilor dintre cuvinte este permisă)