Lucrarea 5

Recursivitate

Atunci când o funcție se apelează pe sine direct sau indirect, spunem că avem recursivitate.

O strategie, care nu dă greş, de definire a funcțiilor recursive ar fi:

- începe prin a scrie condiția de oprire;
- scrie apoi apelul recursiv.

Exemple:

1. Funcția factorial

```
(defun factorial (n)  (if (zerop n) 1 (* n (factorial (- n 1)))))
FACTORIAL
> (factorial 3)
6
```

Sau

```
> (defun factorial (n)  (cond ((zerop n) 1)
          (t (* n (factorial (- n 1))))))
FACTORIAL
> (factorial 4)
24
```

Recursivitate

2. Funcția exponențială:

```
> (defun expt (m n) (cond ((zerop n) 1)
        (t (* m (expt m (- n 1))))))
EXPT
> (expt 2 4)
16
```

3. Funcția reverse:

```
> (defun my-reverse (lista) (cond ((null lista) lista)
        (t (append (my-reverse (cdr lista)) (list (car lista))))))
MY-REVERSE
> (my-reverse '(a b c d e f))
(F E D C B A)
```

4.Funcția de adunare a elementelor unei liste:

```
> (defun suma1 (lst ac)
        (if (null lst) ac (suma1 (cdr lst) (+ ac (car lst)))))
SUMA1
> (suma1 '(1 2 3 4 5) 0)
15
```

Ac – este o variabilă în care calculăm sumele intermediare.

Forme de iterare

Forma **DO** reprezintă maniera cea mai generală de a organiza o iterație în Lisp. Când Dorim să executăm ceva în mod repetat, câteodată este mult mai natural să folosim iterația decât recursivitatea. Ea permite utilizarea unui număr oarecare de variabile şi controlarea valorilor lor de la un pas al iterației la următorul. Forma cea mai complexă a unui apel do este:

```
(DO ((<param 1> <valoare initala 1> <actualizare 1>)
     (<param 2> <valoare initala 2> <actualizare 2>)
     ...
     (<param n> <valoare initala n> <actualizare n>))

     (<test sfarsit> <rezultat>)
     (<corpul lui do>))
```

Primele n elemente ale unui do sunt câte o lisă ce definesc variabilele de control ale buclei, valorile lor inițiale şi actualizarea. Astfel, fiecărei variabile îi corespunde un nume, valoarea ei inițială si o formă de incrementare a acesteia. Dacă expresia de inițializare e omisă, ea va fi implicit considerată nil. Dacă expresia de incrementare este omisă, variabila nu va fi schimbată între paşii consecutivi ai iterației (deşi corpul lui do poate modifica valorile variabilei prin setq).
Înainte de prima iteraţie, toate formele de iniţializare sunt evaluate şi fiecare variabilă este legată la valoarea de iniţializare corespunzătoare (acestea sunt legări iar nu asignări, astfel încât după ieşirea din iteraţie, variabilele revin la valorile la care erau legate înainte de intrarea în iteraţie).

Forme de iterare

Exemple:

1.Funcţia Reverse:

```
> (defun list-reverse(lst)
(do((l lst (cdr l)) (x '() (cons (car l) x)))
((null l) x)))
LIST-REVERSE
```

Sau

```
> (defun list-reverse(lst)
(do((l lst (cdr l)) (x '() ))
((null l) x)
(setq x (cons (car l) x))))
LIST-REVERSE
```

În acest caz, legarea variabilei x cu noua valoare se face în corpul lui DO numai prin SETQ.

```
> (list-reverse '(a b c d e f))
(F E D C B A)
```

Corpul lui DO poate lipsi dacă reprezintă o formă de incrementare a unei variabile iniţializate în DO.

2.Funcţia exponenţială:

```
> (defun expt(m n)
(do((rez 1 (* m rez))
(exp n (- exp 1)))
((zerop exp) rez)))
EXPT
> (expt 3 5)
243
```

Probleme

1.(1p)Să se implementeze recursiv o funcţie COUNTATOMS care primeşte ca argument o listă ce conţine şi liste imbricate şi returnează numărul total de elemente ale listei .

2.(2p)Să se implementeze funcțiile recursive *Rot-left* şi *Rot-right* care rotesc la stânga, respectiv la dreapta n elemente ale unei liste.

Rotire la stânga înseamnă mutarea primului element al listei la coada listei.

Rotire la dreapta înseamnă mutarea ultimului element al listei în fața listei.

3. (1p)Scrieţi o funcţie *fără _dubluri (lista)* care returnează lista fără dubluri. Elementele să fie în aceeaşi ordine în lista returnată ca şi în lista iniţială, păstrându se ultima apariţie a elementelor duplicat.

Exemplu: > (f a r a _d u b l u r i ’ (1 2 2 3 4 5 6 4 4)) → (1 2 3 5 6 4)

4.(1p)Implementați recursiv predicatul *presentp* care determină dacă un atom apare oriunde în interiorul unei liste, chiar şi o listă ce conține liste imbricate.

5.(1p)Implementați recursiv o funcție care primeşte ca argument un număr întreg şi returnează lista cu cifrele numărului.

6.(1p) Implementați iterativ funcția pozpar, care primeşte ca argument o listă şi returnează o listă cu elmentele de pe pozițiile impare ale listei inițiale.

7.(2p)Implementați iterativ o funcție , care determină cel mai mare divizor comun a două numere întregi. Evidenţiaţi parametri actuali şi valoarea rezultată a funcţiei apelate, în ordinea apelurilor, respectiv terminărilor funcţiei pentru un exemplu concret.