CURS 12

Generatori. Argumente opționale. OBLIST și ALIST. Macrodefiniții. Apostroful invers

Cuprins

1. Generatori

Ca exemplu sugestiv pentru această problematică să consideram exemplul unei funcţii **VERIF** care primeşte o listă de liste şi întoarce T dacă toate sublistele de la nivel superficial sunt liniare şi NIL în caz contrar.

(VERIF '((1 2) (a (b))) va produce NIL
(VERIF '((1 2) (a b)) va produce T

Vom folosi două funcții

- Funcția **LIN**, care verifică dacă o listă este liniară

Model recursiv

$$\text{LIN}(l_1 l_2 \dots l_n) = \begin{cases} \text{adevarat} & daca\ l = \emptyset \\ \text{fals} & dacă\ l_1\ nu\ e\ atom \\ \text{LIN}(l_2 \dots l_n) & altfel \end{cases}$$

```
(DEFUN LIN (L)
        (COND
                ((NULL L) T)
                (T (AND (ATOM (CAR L)) (LIN (CDR L) )))
        )
)
```

- Funcția **VERIF**, care verifică dacă o listă de liste are toatele sublistele de la nivel superficial liste liniare.

Model recursiv

$$\text{VERIF}(l_1 l_2 \dots l_n) = \begin{cases} \text{adevarat} & daca\ l = \emptyset \\ \text{fals} & dacă\ \neg \text{LIN}(l_1) \\ \text{VERIF}(l_2 \dots l_n) & altfel \end{cases}$$

```
(DEFUN VERIF (L)
        (COND
                ((NULL L) T)
                (T (AND (LIN (CAR L)) (VERIF (CDR L) )))
        )
)
```

Să observăm că funcţiile LIN şi VERIF de mai sus au aceeaşi structură, conformă şablonului

```
(DEFUN F (L)
        (COND
                ((NULL L) T)
                (T (AND (F1 (CAR L)) (F (CDR L))))
        )
)
```

cu F1 = LIN în cazul F = VERIF şi F1 = ATOM în cazul F = LIN. O astfel de structură este des folosită, "reţeta" ei de lucru fiind: elementele unei liste L sunt transmise pe rând (în ordinea apariţiei lor în listă) unei funcţii F (în cazul de mai sus F1) care le va prelucra. Dacă F întoarce NIL acţiunea se încheie, rezultatul final fiind NIL. Altfel, parcurgerea continuă până la epuizarea tuturor elementelor, caz în care rezultatul final este T. Cum această reţetă se poate aplica pentru orice funcţie F ce realizează prelucrarea într-un anume fel a tuturor elementelor unei liste, apare ideea de a scrie o funcţie **generică** (șablon) GEN ce respecta "reţeta", având doi parametri: funcţia ce va realiza prelucrarea şi lista asupra căreia se va acţiona.

Terminologia adoptată pentru astfel de funcţii (sau similare) este cea de **generatori**. Nu este necesar ca un generator să întoarcă T sau NIL. In funcţie de implementare se pot concepe generatori care să întoarcă, de exemplu, lista tuturor rezultatelor non-NIL ale lui F culese până în momentul încetării acţiunii generatorului (L este vidă sau F întoarce NIL).

Model recursiv

$$\text{GEN}(\text{F}, l_1 l_2 \dots l_n) = \begin{cases} \text{adevarat} & daca\ l = \emptyset \\ \text{fals} & dacă\ \neg \text{F}(l_1) \\ \text{GEN}(\text{F}, l_2 \dots l_n) & altfel \end{cases}$$

```
(DEFUN GEN (F L)
        (COND
                ((NULL L) T)
```

```
                (T (AND (FUNCALL F (CAR L)) (GEN F (CDR L))))
        )
)
```

După ce a fost definit generatorul, se poate defini funcţia VERIF (L) astfel:

```
(DEFUN VERIF (L)
        (GEN #'(LAMBDA (L)
                        (GEN #'ATOM L)
                 )
                L
        )
)
```

2. Argumente opționale

În lista parametrilor formali ai unei funcţii putem folosi următoarele variabile: &OPTIONAL și &REST.

- &OPTIONAL - dacă apare în lista parametrilor formali atunci următorul parametru formal va lua ca valoare parametrul corespunzător din lista parametrilor actuali, respectiv NIL dacă parametrul actual nu există;

- &REST - dacă apare în lista parametrilor formali atunci următorul parametru formal va lua ca valoare lista parametrilor actuali rămaşi neatribuiţi, respectiv NIL dacă nu mai există parametri actuali neatribuiţi.

Exemple

Exemplu 1

```
(defun f (x &optional y)
        (cond
                ((null y)  x)
                (t (+ x y))
        )
)
```

- (f 1 2) → 3
- (f 3) → 3

Exemplu 2

```
(defun g (x &rest y)
```

```
        (+ x (apply #'+ y))
    )
```

- (g 1 2 3) → 6
- (g 4 5) → 9
- (g 3) → 3

Exemplu 3

```
(DEFUN F (L1 &REST L2)
    (COND
        ((NULL (CAR L2)) NIL)
        (T (CONS (MAPCAR #'* L1 (CAR L2)) (F L1 (CADR L2))))
    )
)
```

- (F '(1 2) '(3 4) '(5 6)) se evaluează la ((3 8) (5 12))

- **Exemplu 4**

Să presupunem că dorim să construim o funcţie care să adune 1 la valoarea unei expresii:

```
(DEFUN INC (NUMAR)
    (+ NUMAR 1)
)
```

Această funcţie se va utiliza în felul următor:

- (INC 10) se va evalua la 11

Sigur că, în măsura în care dorim, putem adăuga funcţii asemănătoare şi pentru alte valori de incrementare. Alternativ, putem să rescriem funcţia INC astfel încât să accepte un al doilea parametru:

```
(DEFUN INC (NUMAR INCREMENT)
    (+ NUMAR INCREMENT)
)
```

In marea majoritate a cazurilor, totuşi, vom avea nevoie de incrementarea cu 1 a valorii expresiei corespunzătoare lui NUMAR. In această situaţie putem folosi facilitatea argumentelor opţionale. Iată în continuare definiţia funcţiei INC cu un argument opţional:

```
(DEFUN INC (NUMAR &OPTIONAL INCREMENT)
    (COND
        ((NULL INCREMENT) (+ NUMAR 1))
        (T (+ NUMAR INCREMENT))
    )
)
```

Caracterul & din &OPTIONAL semnalează faptul că &OPTIONAL este un separator de parametri, nu un parametru propriu-zis. Parametrii care urmează după &OPTIONAL sunt legaţi la intrarea în procedură la fel ca şi ceilalţi parametri. Dacă nu există parametru actual corespunzător, valoarea unui parametru formal opţional este considerată NIL. Iată câteva exemple de utilizare a funcţiei INC pe care tocmai am construit-o:

- (INC 10) se evaluează la 11
- (INC 10 3) se evalueaza la 13
- (INC (* 10 2) 5) se evalueaza la 25

Putem, de asemenea, să luăm în considerare o valoare implicită pentru parametrii opţionali. Iată o definiţie a funcţiei INC în care parametrul opţional are valoarea implicită 1:

```
(DEFUN INC (NUMAR &OPTIONAL (INCREMENT 1))
        (+ NUMAR INCREMENT)
)
```

Observaţi că de această dată în locul parametrului opţional este precizată o listă formată din două elemente: primul element este numele parametrului opţional, iar al doilea element este valoarea cu care se iniţializează acesta atunci când argumentul actual corespunzător nu este prezent.

Să notăm că putem avea oricâte argumente opţionale. Toate aceste argumente vor fi trecute în lista parametrilor după simbolul &OPTIONAL, care va apare o singură dată. De asemenea, putem avea un singur argument opţional semnalat prin &REST, a cărui valoare devine lista tuturor argumentelor date cu excepţia celor care corespund parametrilor obligatorii şi opţionali. Fie o nouă versiune a funcţiei INC:

```
(DEFUN INC (NUMAR &REST INCREMENT)
        (COND
                ((NULL INCREMENT) (+ NUMAR 1))
                (T (+ NUMAR (APPLY #'+ INCREMENT)))
        )
)
```

Iată câteva exemple de aplicare a acestei funcţii:

- (INC 10) se evaluează la 11
- (INC 10 1 2 3) se evaluează la 16

3. OBLIST și ALIST

Un obiect Lisp este o structură alcătuită din:

- numele simbolului;
- valoarea sa;
- lista de proprietăţi asociată simbolului.

Gestiunea simbolurilor folosite într-un program Lisp este realizată de sistem cu ajutorul unei tabele speciale numită lista obiectelor (**OBLIST**). Orice simbol este un obiect unic în sistem. Apariţia unui nou simbol determină adăugarea lui la OBLIST. La fiecare întâlnire a unui atom, el este căutat in OBLIST. Dacă se găseşte, se întoarce adresa lui.

Pentru implementarea mecanismului de apel, sistemul Lisp foloseşte o altă listă, numită lista argumentelor (**ALIST**). Fiecare element din **ALIST** este o pereche cu punct formată dintr-un parametru formal şi argumentul asociat sau valoarea acestuia.

In general, o funcţie definită cu *n* parametri P_1, P_2, ..., P_n şi apelată prin (F A_1 A_2 ... A_n) va adăuga la ALIST n perechi de forma (P_i . A_i) daca funcţia îşi evaluează argumentele sau (P_i . A_i'), unde A_i' este valoarea argumentului A_i, daca mediul Lisp evaluează argumentele. De exemplu, pentru funcţia

```
(DEFUN F (A B)
        (COND
                ((ATOM B) A)
                (T (CONS A B))
        )
)
```

apelată cu (F 'X '(1.1)), vom avea pentru ALIST structura (... (A . X) (B . (1 . 1))).

Dacă la momentul apelului simbolurile reprezentând parametrii formali aveau deja valori, acestea sunt salvate în vederea restaurării ulterioare.

Simbolurile sunt reprezentate în structură prin adresa lor din OBLIST, iar atomii numerici şi cei şir de caractere prin valoarea lor direct în celula care îi conţine.

Iniţial, la începutul programului, ALIST este vidă. Pe măsura evaluării de noi funcţii, se adaugă perechi la ALIST, iar la terminarea evaluării funcţiei, perechile create la apel se şterg. In corpul funcţiei în curs de evaluare valoarea unui simbol *s* este căutată întâi în perechile din ALIST începând dinspre vârf spre bază (această acţiune este cea care stabileşte dinamic domeniul de vizibilitate). Dacă valoarea nu este găsită în ALIST se continuă căutarea în OBLIST. Aşadar, este clar că **ALIST şi OBLIST formează contextul curent al execuţiei unui program.**

Exemplu

- (SETQ X 1) iniţializează simbolul X cu valoarea 1;
- (SETQ Y 10) iniţializează simbolul Y cu valoarea 10;
- (DEFUN DEC (X) (SETQ X (- X 1))) defineşte o funcţie de decrementare a valorii parametrului;
- (DEC X) se evaluează la 0;
- X se evaluează tot la 1;
- (DEC Y) se evaluează la 9;
- Y se evaluează tot la 10.

Să observăm deci că modificările operate asupra valorilor simbolurilor reprezentând argumentele formale se vor pierde după ieşirea din corpul funcţiei şi revenirea în contextul apelator.

4. Macrodefiniții

Din punct de vedere sintactic, macrodefiniţiile se construiesc în acelaşi mod ca funcţiile, cu diferenţa că în loc să se folosească funcţia DEFUN se va folosi funcţia DEFMACRO:

(DEFMACRO s l f1 ... f n): s

Funcţia DEFMACRO creează o macrodefiniţie având ca nume primul argument (simbolul s), iar ca parametri formali elementele simboluri ale listei ce constituie al doilea argument; corpul macrodefiniţiei create este alcătuit din una sau mai multe forme aflate, ca argumente, pe a treia poziţie şi eventual pe următoarele. Valoarea întoarsă ca rezultat este numele macrocomenzii create. Funcţia DEFMACRO nu-şi evalueaza niciun argument.

- evaluarea argumentelor face parte din procesul de evaluare al macrodefiniției

Modul de lucru al macrodefiniţiilor create cu DEFMACRO este diferit de cel al funcţiilor create cu DEFUN:

- parametrii macrodefiniţiei nu sunt evaluaţi;
- se evaluează corpul macrodefiniţiei şi se va produce o S-expresie intermediară;
- această S-expresie va fi evaluată, acesta fiind momentul în care sunt evaluaţi parametrii.

Să vedem mai întâi un exemplu comparativ. Fie următoarele definiţii și evaluări:

> (DEFUN F (N) (PRINT N)) > (SETQ N 10) > (F N) 10 10	> (DEFMACRO G (N) (PRINT N)) > (SETQ N 10) > (G N) N 10
Notă Parametrul N este evaluat de la bun început, PRINT va tipări valoarea acestuia, şi o va întoarce ca valoare a apelului funcţiei.	**Notă** În primul moment nu se face o încercare de a se evalua parametrul N. Ca atare, PRINT va tipări N, după care îl va întoarce pe N ca valoare a formei intermediare a macrodefiniţiei. Apoi această valoare intermediară este evaluată şi se va produce 10.

Ca un alt exemplu, dorim să producem o macrodefiniţie DEC care să decrementeze cu 1 valoarea parametrului ei, ca în exemplul următor:

- (SETF X 10) se evaluează la 10
- (DEC X) va produce 9
- X se evaluează la 9

Aceasta înseamnă că forma intermediară va trebui să fie

(SETQ parametru (- parametru 1))

Iată o posibilitate:

```
(DEFMACRO DEC (N)
        (LIST 'SETQ N (LIST '- N 1))
)
```

5. Apostroful invers (backquote)

Apostroful invers (`) uşurează foarte mult scrierea macrocomenzilor. Oferă o modalitate de a crea expresii în care cea mai mare parte este fixă, şi în care doar câteva detalii variabile trebuie completate. Efectul apostrofului invers este asemănător cu al apostrofului normal (′), în sensul că blochează evaluarea S-expresiei care urmează, cu excepţia că orice **virgulă** (**,**) care apare va produce *evaluarea* expresiei care urmează. Iată un exemplu:

- (SETQ V ′EXEMPLU)
- `(ACESTA ESTE UN ,V) se evaluează la (ACESTA ESTE UN EXEMPLU)

De asemenea, apostroful invers acceptă construcția **,@**. Această combinaţie produce şi ea evaluarea expresiei care urmează, dar cu diferenţa că valoarea rezultată trebuie să fie o listă. Elemente acestei liste sunt dizolvate în lista în care apare combinaţia **,@**. Iată un exemplu:

- (SETQ V ′(ALT EXEMPLU))
- `(ACESTA ESTE UN ,V) se evaluează la
  (ACESTA ESTE UN (ALT EXEMPLU))
- `(ACESTA ESTE UN ,@V) se evaluează la
  (ACESTA ESTE UN ALT EXEMPLU)

Folosind apostroful invers, macrodefiniţia DEC se va putea scrie mai simplu astfel:

```
(DEFMACRO DEC (N)
        `(SETQ ,N (- ,N 1))
)
```