Formă în LISP

→ listă evaluabilă

↓

adică listă în care primul ei element este numele unei funcții

Obs 1. O funcție este mereu evaluată dacă nu are QUOTE !!

Obs 2. Primul argument al oricărei liste NU este niciodată evaluat !!

⇒ LISP nu evaluează primul element dintr-o formă, ci doar îl aplică !!

- SET

(SET s_1 R_1 s_2 R_2 ... s_n R_n)

→ se evaluează și s_1 .. s_n și R_1, ... R_n

→ valorile simb. evaluate s_1, .. s_n

ex. (SET 'x 'A) ; x se eval. la A

(SET x 'B) ; A se eval. la B

(SET 'x (CONS (SET 'y x) '(B)))

y se eval. la A

(CONS 'A '(B)) = '(A B)

x se evaluează la (A B)

! întoarce rezultatul evaluării lui R_n

SETQ

(SETQ s_1 R_1 ... s_n R_n)

→ se evaluează doar R_1, .. R_n

→ arg. s_1, .. s_n nevaluate vor primi ca valori R_1, .. R_n

! întoarce rezultatul evaluării lui R_n

ex. (SETQ x 'A) = A

(SETQ A '(B C)) = (B C)

(CAR A) = C

(SETQ x (CONS x A)) = (A B C)

x se eval. la (A B C)

SETF

(SETF A_1 R_1 .. s_n R_n)

! efect destructiv

→ s_1 ... s_n forme care în momentul evaluării macroului accesează un obiect Lisp

valorile lor se leagă de ~~valorile~~ formelor R_1', .. R_n' evaluate

! întoarce rez. evaluării lui R_n'

ex. (SETQ A '(B C D)) ; A se eval. la (B C D)

(SETF (CADR A) 'x) ; A se eval. la (B x D)

(SET (CADR A) 'y) ; x se inițializează la y

↓

x

A : (B x D)

x : y

Atenție !!

(SETQ Q 'CAR)

(Q '(ABC)) → greșit ! Primul arg. din listă nu se evaluează !!

((EVAL Q) '(ABC)) → tot greșit !!

Soluție: Apply și Funcall

(EXPR_FUNC $l_1 \dots l_n$)

↓

evaluată până a se obține funcție / expresie ce poate fi aplicată parametrilor

(APPLY KF l_A) : e

→ parametri trebuie furnizați sub formă de listă!

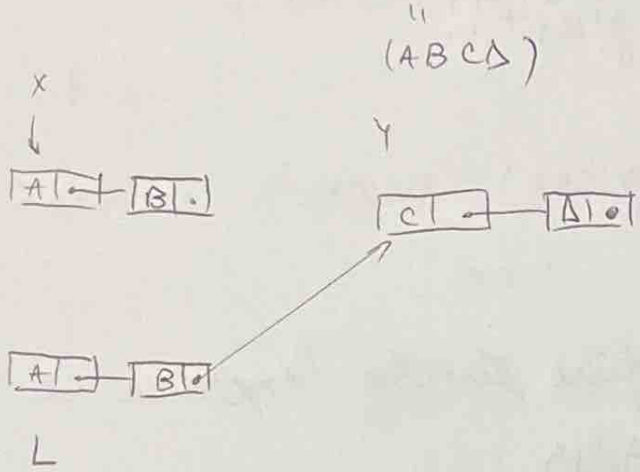
→ evaluează KF și aplică funcția parametrilor din listă

(FUNCALL KF $l_A \dots$)^{l_n} : e

→ la fel ca apply, doar că sunt numărat fix de parametri!

Atenție la APPEND!

```
(SETQ x '(A B))   ; x se eval. la (A B)
(SETQ y '(C D))   ; y se eval. la (C D)
(SETQ L (APPEND x y))   ; L se eval. la (A B C D)
```




```
(SETF (CAR x) 'E)   ; x se ev. la (E B)
L → (A B C D)
(SETF (CADR y) 'F)   ; y se ev. la (C F)
L → (A B C F)
```

EVAL

(EVAL e) : e

→ se evaluează forma e
→ se returnează rezultatul evaluării

1. Se evaluează e
2. Rezultatul evaluării este evaluat ca efect al aplicării EVAL

ex. (SETQ L '(1 2 3))
(SETQ P '(CAR L))
(EVAL P) → 1
↓
(CAR L)

① (DEFUN G(L)

(+ (CAR L)(CADR L))

)

→ oprește evaluarea

(SETQ H 'F) ; H se evaluează la F

(SET H 'G) ; H se evaluează la ~~F~~

↓ se evaluează

~~F~~ se evaluează la (+ (CAR L) (CADR L))

~~(F '(2 3 4 5 6)) → 2+3=5~~

Eroare undefined function F

(F '(2 3 4 5 6)) !! Primul arg. dintr-o listă nu se evaluează

Răspuns: (funcall #' '(2 3 4 5 6))

② (SET 'L '(T NIL T))

(APPLY 'OR L) → eroare pt. că OR este operator logic! NU funcție !!

Soluție:

```
(defun sau (l)
    (cond
        ((null l) nil)
        (t (OR (car l) (cdr l)))
    )
)
```

→ (apply #'sau list(l))

→ (~~apply~~ funcall #'sau l)

APPEND pe L,
(CAR L) ....

③ (DEFUN G(L)
(MAPCON #'LIST L)
)

(APPLY #'APPEND (MAPCON #'G '(12)))

(12) (2)

~~((12) (2))~~ ~~((2))~~

~~((12) (2) (2))~~

(1 2 2) (2)

(1 2 2 2)

MAP LIST → (L) , (CDR L) ....
→ rezultatele grupate cu LIST

④ (defun g(l)
(list (car l)(car l))
)

(SETQ Q 'g) ; Q se evaluează la g
(SETQ P Q) ; P se evaluează la g
↓
g

(funcall p '(A B C)) → list A A → (A A)

③ (F '(1 2)) → (1 2)

(F '(3 4) '(5 6) '(7 8))

x → y

APPEND '(3 4) '(5 7) → (3 4 5 7)

MAPCAR #'CAR '('(6 6) '(7 9))

(5 7)

④ (SET H 'F) ; H ??
eroare ??

(SETQ H 'F) ; H evaluează la P
(SETQ H 'G) ; P se evaluează la G

⑤ (SETQ F 10) ; F se evaluează la 10
(SETQ G 'F) ; G se evaluează la F nu la 10!!
(G '(0 1 2 3 7 9)) → nu merge! A. că 'oprește evaluarea
(apply funcall G '(0 1 2 3 7 9))

⑥ (defun G(l)
(* (car l)(cdr l))
)
(SETQ CAR 'G) ; CAR se evaluează la G
(CAR '(2 3 5 6)) ; → !!
↳ nu se evaluează !!
CAR RĂMÂNE CAR (funcție de sistem !!)

⑦ p([], 0).

p([H|T], S) :- !, H mod 2 =:= 0, p(T, S1), S is S1*H.

p([_|T], S) :- p(T, S1), S is S1.

p([1,2,3,4,5], S) → false, pt. că se intră pe clauza a 2-a, ! oprește căutarea la predicatul următor.

(

⑧ (defun inc (x)
(+ 1 x)
)

(SETQ A 1) ; A se evaluează la 1

(INC A) ; se ret. rezultatul formei (+1 x) adică 2

! valoarea variabilei A nu se modifică !! Ne-ar trebui o macrodefinire