

CURS 11

Exemple funcții MAP (cont).

EXEMPLU 2.1 Fie următoarele definiții

```
(defun f (L e)
  (list e L)
)
```

```
(setq L '(1 2 3))
(setq e 4)
```

(mapcar #'f L e) se evaluează la NIL
(mapcar #'(lambda (L) (f L e)) L) se evaluează la ((4 1) (4 2) (4 3))

EXEMPLU 2.2 Fie următoarea definiție de funcție

```
(defun f (L)
  (list L)
)
```

(mapcar #'f '(1 2 3)) se evaluează la ((1) (2) (3))
(mapcan #'f '(1 2 3)) se evaluează la (1 2 3)

De remarcat echivalența între

(mapcan #'f L) și (apply #'append (mapcar #'f L))

EXEMPLU 2.3 Să se definească o funcție care să returneze lungimea unei liste neliniare (în număr de atomi la orice nivel)

$(LG '(1 (2 (a) c d) (3))) = 6$

$$LG(L) = \begin{cases} 1 & \text{daca } L \text{ e atom} \\ \sum_{i=1}^n LG(L_i) & \text{daca } L \text{ e lista } (L_1 \dots L_n) \end{cases}$$

```
(DEFUN LG (L)
  (COND
    ((ATOM L) 1)
    (T (APPLY #'+ (MAPCAR #'LG L)))
  )
)
```

EXEMPLU 2.4 Să se definească o funcție care având ca parametru o listă neliniară să returneze numărul de subliste (inclusiv lista) având lungime număr par (la nivel superficial).

$(nr '(1 (2 (3 (4 5) 6)) (7 (8 9)))) = 4$

Vom folosi o funcție auxiliară care returnează T dacă lista argument are număr par de elemente la nivel superficial, NIL în caz contrar.

$$nr(L) = \begin{cases} 0 & \text{daca } L \text{ e atom} \\ 1 + \sum_{i=1}^n lg(L_i) & \text{daca } L \text{ e lista } (L_1 \dots L_n) \text{ si } n \text{ e par} \\ \sum_{i=1}^n lg(L_i) & \text{altfel} \end{cases}$$

```
(DEFUN PAR (L)
  (COND
    ((= 0 (MOD (LENGTH L) 2)) T)
    (T NIL)
  )
)

(DEFUN nr (L)
  (COND
    ((ATOM L) 0)
    ((PAR L) (+ 1 (APPLY #'(MAPCAR #'nr L))))
    (T (APPLY #'(MAPCAR #'nr L))))
  )
)
```

EXEMPLU 2.5 Să se definească o funcție care având ca parametru o listă neliniară să returneze lista atomilor (de la orice nivel) din listă.

$(atomi '(1 (2 (3 (4 5) 6)) (7 (8 9)))) = (1 2 3 4 5 6 7 8 9)$

$$atomi(L) = \begin{cases} (L) & \text{daca } L \text{ e atom} \\ \bigcup_{i=1}^n atomi(L_i) & \text{daca } L \text{ e lista } (L_1 \dots L_n) \end{cases}$$

```
(DEFUN atomi (L)
  (COND
    ((ATOM L) (LIST L))
    (T (MAPCAN #'(atomi L))))
  )
)
```

Observație: Varianta echivalentă folosind funcția MAPCAR

```
(DEFUN atomi (L)
  (COND
    ((ATOM L) (LIST L))
    (T (APPLY #'APPEND (MAPCAR #'atomi L)))
  )
)
```

EXEMPLU 2.6 Să se definească o funcție care determină numărul de apariții, de la orice nivel, ale unui element într-o listă neliniară.

(nrap 'a '(1 (a (3 (4 a) a)) (7 (a 9)))) = 4

$$nrap(e, l) = \begin{cases} 1 & \text{daca } l = e \\ 0 & \text{dacă } l \text{ e atom} \\ \sum_{i=1}^n nrap(e, l_i) & \text{altfel, } l = (l_1 l_2 \dots l_n) \text{ e lista} \end{cases}$$

```
(defun nrap(e L)
  (cond
    ((equal L e) 1)
    ((atom L) 0)
    (t (apply #'+ (mapcar #'(lambda(L)
                                (nrap e L))
                          L)
      )
    )
  )
)
```

EXEMPLU 2.7 Se dă o listă neliniară. Se cere să se returneze lista din care au fost șterși atomii numerici negativi. Se va folosi o funcție MAP.

Ex: (stergere '(a 2 (b -4 (c -6)) -1)) → (a 2 (b (c)))

$$sterg(l) = \begin{cases} \emptyset & \text{daca } l \text{ numeric negativ} \\ l & \text{dacă } l \text{ e atom} \\ \bigcup_{i=1}^n sterg(l_i) & \text{altfel, } l = (l_1 l_2 \dots l_n) \text{ e lista} \end{cases}$$

```

(defun sterg(L)
  (cond
    ((and (numberp L) (minusp L)) nil)
    ((atom L) (list L))
    (t (list (apply #'append
                    (mapcar #'sterg L)
                    ; (sterg '((a) (c -2))) → ((a) (c))
                    ; (sterg '(a) → (a)
                    ; (sterg '(c -2) → (c)
                    ; (append '(a) '(c)) → (a c)
                    ; (append '((a)) '((c))) → ((a) (c))
                )
            )
      )
    )
  )
)

(defun stergere(L)
  (car (sterg L))
)

```

EXEMPLU 2.8 Se dă un arbore n-ar nevid, reprezentat sub forma unei liste neliniare de forma (rădăcina lista_sub1.....lista_sub_n) (V1 de reprezentare a arborilor binari, Curs 9). Se cere să se determine numărul de noduri din arbore.

(nrNoduri '(a (b (c) (d (e))) (f (g)))) va produce 7

Model recursiv

$$nrNoduri(l_1 l_2 \dots l_n) = \begin{cases} 1 & \text{daca } n = 1 \\ 1 + \sum_{i=2}^n nrNoduri(l_i) & \text{altfel} \end{cases}$$

```

(defun nrNoduri(L)
  (cond
    ((null (cdr L)) 1)
    (t (+ 1 (apply #'+
                  (mapcar #'nrNoduri (cdr L))
                )
      )
    )
  )
)

```

EXEMPLU 2.9 Se dă un arbore n-ar nevid, reprezentat sub forma unei liste neliniare de forma (rădăcina lista_sub1.....lista_sub_n) (V1 de reprezentare a arborilor binari, Curs 9). Se cere să se determine adâncimea arborelui. Observație. Nivelul rădăcinii este 0.

(adancime '(a (b (c) (d (e))) (f (g)))) va produce 3

Model recursiv

$$adancime(l_1 l_2 \dots l_n) = \begin{cases} 0 & \text{daca } n = 1 \\ 1 + \max(adancime(l_2), adancime(l_3), \dots, adancime(l_n)) & \text{altfel} \end{cases}$$

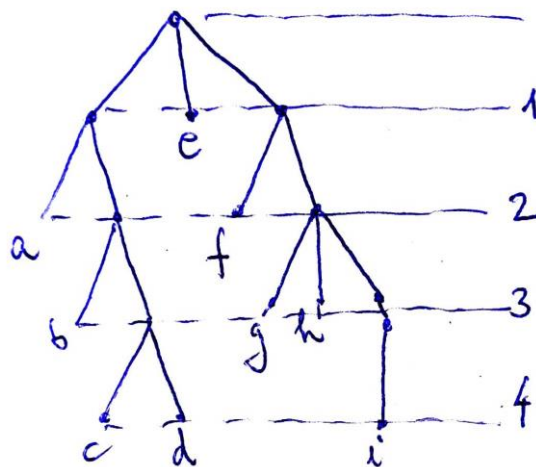
```
(defun adancime(L)
  (cond
    ((null (cdr L)) 0)
    (t (+ 1 (apply #'max
                    (mapcar #'adancime (cdr L))
                    )
        )
    )
  )
)
```

EXEMPLU 2.10 Să se determine lista atomilor de adâncime n dintr-o listă neliniară (nivelul superficial al listei se consideră 1).

(lista '((a (b (c d))) e (f (g h (i)))) 3) va produce (b g h)

(lista '((a (b (c d))) e (f (g h (i)))) 4) va produce (c d i)

(lista '((a (b (c d))) e (f (g h (i)))) 5) va produce NIL



Model recursiv

$$lista(l, n) = \begin{cases} (l) & \text{dacă } n = 0 \text{ și } l \text{ atom} \\ \emptyset & \text{dacă } n = 0 \\ \emptyset & \text{dacă } l \text{ atom} \\ \bigcup_{i=1}^k lista(l_i, n-1) & \text{altfel, } l = (l_1 l_2 \dots l_k) \text{ e lista} \end{cases}$$

```
(defun lista(L n)
  (cond
    ((and (= n 0) (atom L)) (list L))
    ((= n 0) nil)
    ((atom L) nil)
    (t (mapcan #'(lambda(L)
                     (lista L (- n 1)))
                 L)
        )
    )
  )
)
```

!!! Clauza 4 din COND poate fi rescrisa cu MAPCAR

```
(t (apply #'append (mapcar #'(lambda(L)
                                (lista L (- n 1)))
                                L)
    )
)
```

EXEMPLU 2.11 Se dă o listă de liste. Se cere să se determine

```
(defun m (L)
  (cond
    ((numberp L) L)
    ((atom L) most-negative-fixnum)
    (t (apply #'max
               (mapcar #'m L)
               )
        )
    )
  )
)

(defun lista (L)
  (mapcan #'(lambda (L)
              ((lambda (v)
                 (cond
                   ((= 0 (mod v 2)) (list v))
```


(fct '((1 2) (4 5) (7 8))) → ((1 4 7) (2 5 8))

EXEMPLU 2.14 Să se definească o funcție care având ca parametru o listă neliniară returnează lista liniară a atomilor care apar pe orice nivel, dar în ordine inversă.

(INVERSARE '(A (B C (D (E))) (F G))) = (G F E D C B A)

```
(DEFUN INVERSARE (L)
  (COND
    ((ATOM L) (LIST L))
    (T (APPLY #'APPEND (MAPCAR #'INVERSARE (INVERS L))))
  )
)
```

Unde INVERS este funcția care returnează o listă inversată la nivel superficial (Curs 8, Exemplu 4.3) – echivalent cu apelul funcției REVERSE (funcție predefinită).

EXEMPLU 2.15 O matrice se poate reprezenta în Lisp sub forma unei liste ale cărei elemente sunt liste reprezentând liniile matricei.

((linia1) (linia2)...)

Să se definească o funcție care având ca parametri două matrice de ordin n returnează (sub formă de matrice) produsul acestora.

(PRODUS '((1 2) (3 4)) '((2 -1) (3 1))) = ((8 1) (18 1))

Observație: Vom folosi două funcții ajutătoare: o funcție (COLOANE L) care returnează lista coloanelor matricei parametru L și o funcție (PR L1 L2) care returnează sub formă de matrice rezultatul înmulțirii matricei L1 (listă de linii) cu lista L2 (o listă de coloane ale unei matrice).

```
(DEFUN COLOANE (L)
  (COND
    ((NULL (CAR L)) NIL)
    (T (CONS (MAPCAR #'CAR L) (COLOANE (MAPCAR #'CDR L))))
  )
)
```

```
(DEFUN PR (L1 L2)
  (COND
    ((NULL (CAR L1)) NIL)
    (T (CONS (MAPCAR #'(LAMBDA (L)
      (APPLY #'+ (MAPCAR #'* (CAR L1) L))
    )
      L2)
      (PR (CDDR L1) L2)
    )
  )
)
```



```

                                L2
                                )
                                (PR (CDR L1) L2)
                            )
                        )
                    )
                )
            )
        )
    )
(DEFUN PRODUS (L1 L2)
  (PR L1 (COLOANE L2))
)

```

EXEMPLU 2.16 Se dă o mulțime reprezentată sub forma unei liste liniare. Se cere să se genereze lista submulțimilor mulțimii. Se va folosi o funcție MAP.

Ex: (*subm* '(1 2)) → (nil (1) (2) (1 2))

Observație

(setq e 1)

(mapcar #'(lambda(L) (cons e L)) '((2 3) (3 2))) va produce ((1 2 3) (1 3 2))

```

(defun subm (L)
  (cond
    ((null L) (list nil))
    (t ((lambda (s)
          (append s (mapcar #'(lambda (sb)
                                (cons (car L) sb)
                                )
                            s)
          )
        )
      (subm (cdr L))
    )
  )
)

```

EXEMPLU 2.17 Se dă o mulțime reprezentată sub forma unei liste liniare. Se cere să se genereze lista permutărilor mulțimii. Se va folosi o funcție MAP.

Ex: (*permutari* '(1 2 3)) → ((1 2 3) (1 3 2) (2 1 3) (2 3 1) (3 1 2) (3 2 1))

```

(defun permutari (L)
  (cond
    ((null (cdr L)) (list L))
    (t (mapcan #'(lambda (e)

```

unde REMOVE este funcția care returnează lista din care s-a șters un element de la nivel superficial.