

1. Funcția INC are scopul de a incrementa un nr.  
 $(\text{setg } A \_1) \rightarrow A$  se va evalua la 1  
 $(\text{INC } A)$  → se va evalua la 2, decarece A-ul se va evalua la 1 și Funcția INC are scopul de a incrementa parametrul dat

R12

3. Aranjamente de la elemente cu suma S.

Idee a algoritmului:

1. un predicat medeterminist ce determină toate combinațiile valide, funcțional astfel:

- dacă mai trebuie adăugat un singur element și acesta este egal cu suma rezmasă (necesară ca sumă tuturor elem. să fie S), atunci se determină lista de acest element
- se poate să se înceapă peste elementul curent
- dacă vom să adăugăm elem. curent la aranjament (cum în care-l vom adăuga, pe rând, pe toate pozițiile, folosind un predicat auxiliar), trebuie să mai existe loc, iar val. să să nu fie mai mare decât suma rezmasă

2. un predicat mediet. ce inseră un elem., pe rând, pe toate pozițiile unei liste

3. un predicat ce returnează o listă cu toate aranjările valide determinante

Modelle matematice:

înserează ( $e_1 \dots e_m, e$ ) =

1.  $e \oplus e_1 \dots e_m$

2.  $e_1 \oplus$  înserează ( $e_2 \dots e_m, e$ ),  $m > 1$

cremăj ( $e_1 \dots e_m, p_k, s$ ) =

1.  $\{e_1\}$ ,  $p_k=1$  și  $s = e_1$

2.  $\text{cremăj } (e_2 \dots e_m, p_k, s)$

3. înserează (cremăj ( $e_2 \dots e_m, p_{k-1}, s - e_1$ ),  $e_1$ ), dacă

$p_k > 1$  și  $e_1 \in s$

main ( $e, p_k, s$ ) =  $\bigcup \text{cremăj } (e, p_k, s)$

% înserează ( $L$ -lista,  $E$ -element,  $R$ -Lista)

% model de flux: ( $i, i', \alpha$ )  $\rightarrow$  me determinist

înserează ( $L, E, [E | L]$ ). % caz în care se înserează element.

înserează ( $L H | T S, E, R$ ). - % pe prima poziție

înserează ( $T, E, R_1$ ),

$R_1 = [H | R, T]$ .

% cremăj ( $L$ -lista,  $K$ -Nr,  $S$ -Nr,  $R$ -Lista)

%  $R \rightarrow$  nr. de elemente care mai pot fi adăugate în cremăj

%  $S \rightarrow$  suma ramasă

% model de flux: ( $i, i', i'', \alpha$ )  $\rightarrow$  me determinist

cremăj ( $[H | -], i \rightarrow H, [ - ]$ ). % caz în care s-a det. un ramaj valid

cremăj ( $[ - | T S ], K, S, R$ ):-

% care în care se sărăpește elementul current

aranj $\downarrow$ (T, K, S, R).

aranj $\downarrow$ ( $\sum HITS$ , K, S, R) :-

K > 1,

S > 4,

% se adaugă elem. current la aranjament

K<sub>1</sub> is K - 1,

S<sub>1</sub> is S - 1,

aranj $\downarrow$ (T, K<sub>1</sub>, S<sub>1</sub>, R<sub>1</sub>),

inserare(R<sub>1</sub>, H, R).

% maxim(L - lista, K - Nr, S - Nr, R - lista)

% modele de flux: (l, i, r, d)  $\rightarrow$  de terminat

% reunește toate aranjamentele valide într-o listă

maxim(L, K, S, R) :-

fundare(R<sub>1</sub>, aranj $\downarrow$ (L, K, S, R<sub>1</sub>), R).

5. Nr. de subliste pt. care primul element este numeric

Idee algoritm:

1. o funcție ce determină primul element dintr-o listă  
eliminând

2. o funcție ce numără sublistele valide este:

- dacă primul este numeric  $\rightarrow$  o listă ia în considerare
- dacă este o listă ce respectă prop  $\rightarrow$  se adaugă 1 + se parcurg toate sublistele acesteia
- altfel, doar se parcurg subliste

Modele matematice.

$$\text{prim}(e_1 \dots e_n) = \begin{cases} e_1, e_1 \text{ è atorn} \\ \text{prim}(e_1), \text{ altfel} \end{cases}$$

$$\text{succiste}(e) = \begin{cases} 0, e \text{ è atorn} \\ 1 + \sum_{i=1}^n \text{succiste}(e_i), \text{ prim}(e) \text{ nu è nro.} \\ \sum_{i=1}^n \text{succiste}(e_i), \text{ altfel} \end{cases}$$

; prim (e - lista)

; restituendo la prima e atorn

(defun prim (e))

(cond

(atorn (car e)) (car e))

(+ (prim (cdr e))))

)

; succiste (e - lista (Atorn))

(defun succiste (e))

(cond

(atorn e) 0)

(not (numerop (prim e))))

; se aggiunge 1 + se race somma de restue wor. date

; di apicarza funzione si succiste per lista' attuale

(+ 1 (apply #'(lambda (mapcar #'succiste e))))

(+ (apply #'(lambda (mapcar #'succiste t))))

)

)