

Liste in PROLOG

Definitie:

Lista in Prolog este o structura de date de tipul :

- (1) lista vida (se noteaza []);
- (2) lista cu un singur element si coada vida (notandu-se [X] sau [X|[]]);
- (3) lista cu un prim element (head) si coada de tip (1),(2) sau (3), notandu-se [H|Tail].

Handler-ele uzuale catre componentele si sublistele unei liste sunt:

Notatie:	Handler catre:	Conditie:
[H _]	primul element	lista nevida
[X,Y _]	primele doua elemente	lista de cel putind doua elemente
[_ Tail]	coada	lista cu cel putin un element
[H Tail]	cap si coada	lista cu cel putin un element

Un exemplu elementar de prelucrare unei liste este calculul lungimii sale (a se vedea secventele de cod postate on-line). In principiu, orice prelucrare a unei liste este la fel de simpla ca parcurgerea sa.

O exceptie interesanta este concatenarea a doua liste:

lconcat([],L,L).

lconcat([H|T],L,[H|R]):-lconcat(T,L,R).

Exercitii:

1. Dublati elementele unei liste. Extrageți sublistele elementelor de indice par/impar. Extrageți sublista elementelor de indice 3k.
2. Reversati o lista numerica. Testati daca o lista este palindrom. Generati o lista de n liste numerice vide.
3. Insumati elementele unei liste numerice. Implementati concatenarea a doua liste. Comentati polimorfismul predicatului.
4. Implementati:
 - un predicat care sa testeze apartenenta unei valori la o lista numerica;
 - un predicat care sa returneze numarul de ocurente a unei valori intr-o lista.
5. Implementati un predicat care sa determine minimul/maximul unei liste numerice. Eliminati dublurile dintr-o lista numerica. Eliminati dublurile consecutive dintr-o lista numerica. Implementati compresia/decompresia Run-Length (Encoding) a unei liste.
6. Se da lista $[0,1,2,3,\dots,2^k-1]$. Calculati lista elementelor obtinute prin reversarea reprezentarilor binare pe k biti a elementelor listei date (<http://fmi.spiruharet.ro/bodorin/articles/brdfpvd.ro.pdf>).
7. Anterior, s-a dat spre studiu problema generarii unei aproximatii fine a numarului lui Euler, limita a sirurilor:

$$(1+1/n)^n, \left(1+\frac{1}{2^k}\right)^{2^k}, \sum_{n=0}^{\infty} \frac{1}{n!}.$$

Se poate constata (si sper ca s-a constatat) ca al treilea sir converge cel mai rapid. Tot cu aceasta ocazie s-a putut identifica domeniul util al factorialului (domeniul de computabilitate) ca fiind 1-170 (impus de codomeniul computational maximal 1 - 1E+308). S-a vazut deci ca aproximarea cea mai buna a lui e poate fi gasita in cel mult 170 de iteratii, fiecare iteratie presupunand o inmultire (pentru actualizarea factorialului), o impartire (1/n!) si o adunare (in total, maxim 510 operatii).

S-a vazut de asemenea ca, folosind cel de al treilea sir, primele **16** zecimale se stabilizeaza in **17** iteratii (**51** operatii aritmetice), iar folosind primul sir, primele **5** zecimale se stabilizeaza in **1.000.000** de iteratii.

Este acum usor de inteles de ce al treilea sir este cel utilizat in toate limbajele si mediile de programare pentru calculul bazei functiei exponentiale.

Creinte:

Cercetati convergenta numerica a seriilor:

$$\sin(x) = \sum_{n=0}^{\infty} \frac{(-1)^n x^{2n+1}}{(2n+1)!}, \quad \cos(x) = \sum_{n=0}^{\infty} \frac{(-1)^n x^{2n}}{(2n)!},$$

implementand predicatle corespunzatoare.

Cate iteratii sunt maxim posibile asumand limita maxima a domeniului real la 1E+308?

Dupa cate iteratii se stabilizeaza primele 16 zecimale?

Punctaj: toate cu un punct si se adauga: 1 punct pentru (6), 2 puncte pentru (7).
