Laboratorul 2 - Programare Logică și Funcțională

Seria 36

Martie 2024

1 Liste

Listele în Prolog sunt un tip special de date. Ele se scriu între paranteze drepte, cu elementele despărţite prin virgulă. Notăm cu `[]` lista vidă.

Exemple de liste în Prolog:

- `[elephant, horse, donkey, dog]`
- `[elephant, [], X, parent(X, ton), [a,b,c], f(22)]`

Întrucât în Prolog nu avem acces la *for loops*, singura modalitate prin care putem efectua iterativ operaţii pe liste este *principiul recursivităţii*. În acest sens, pentru fiecare listă avem două părţi importante, şi anume capul listei (numit *head*) şi coada listei (numită *tail*).

- primul element al unei liste se numeşte *head*, iar restul listei se numeşte *tail*;
- o listă vidă nu are un prim element;
- avem în Prolog o notaţie utilă pentru liste, utilizând operatorul `|`.

Exemple:

```
?- [1, 2, 3, 4, 5] = [Head | Tail].
Head = 1
Tail = [2, 3, 4, 5]
```

Utilizând această notaţie, putem să returnăm uşor, de exemplu, al doilea element dintr-o listă:

```
?- [quod, licet, iovi, non, licet, bovi] = [_, X | _].
X = licet
```

În general, o structură pentru prelucrarea recursivă a listelor este următoarea:

```
my_predicate([], ...) :- base_case(...);
my_predicate([H | T], ...) :- ..., my_predicate(T, ...), ... .
```

2 Exerciţii

2.1 Exerciţiul 1 - recapitulativ

Implementaţi un predicat `fib/2` care primeşte ca prim argument un număr natural `X` şi returnează al `X`-lea termen din şirul lui Fibonacci.

Vă răspunde programul la interogarea de mai jos?

```
?- fib(50, Res).
```

2.2 Exerciţiul 2

Scrieţi un predicat `list_length/2` care determină lungimea unei liste primită ca prim argument.

```
?- list_length([1,2,[]], X).
X = 3
```

2.3 Exerciţiul 3

Scrieţi un predicat `list_sum/2` care să calculeze suma elementelor din lista primită ca prim argument.

```
?- list_sum([1,2,3], X).
X = 6
```

2.4 Exerciţiul 4

Scrieţi un predicat `elements_of/2` care verifică dacă primul argument este element al listei din cel de-al doilea argument.

Exemple de apel şi rezultatele aşteptate:

```
?- elements_of(1, [1,2,3]).
true

?- elements_of(b, [a,b,c]).
true

?- elements_of(d, [a,b,c]).
false

?- elements_of(X, [a,b,c]).
X = a ;
X = b ;
X = c ;
false
```

2.5 Exerciţiul 5

Definiţi un predicat `all_a/1` care primeşte ca argument o listă şi care verifică dacă argumentul său este format doar din `a`-uri. Scrieţi şi o formă generală, care să verifice că toate elementele sunt egale cu un simbol dat. **Ce tip de egalitate utilizaţi?**

```
?- all_a([a,a,a,a]).
true

?- all_a([a,A,a,a]).
A = a
```

2.6 Exerciţiul 6

Scrieţi un predicat `trans_a_b/2` care translatează o listă de `a`-uri într-o listă de `b`-uri. Predicatul trebuie să fie adevărat dacă primul argument este o listă de `a`-uri iar al doilea este o listă de `b`-uri de aceeaşi lungime.

```
?- trans_a_b([a,a,a], L).
L = [b,b,b]

?- trans_a_b([a,a,a], [b]).
false
```

2.7 Exerciţiul 7

Scrieţi un predicat `scalarMult/3` al cărui prim argument este un întreg, al doilea argument este o listă de întregi, iar al treiela argument este rezultatul înmultirii cu scalari al celui de-al doilea argument cu primul.

```
?- scalarMult(3, [2,7,4], Result).
Result = [6,21,12]
```

2.8 Exerciţiul 8

Scrieţi un predicat `dot/3` al cărui prim argument este o listă de întregi, al doilea argument este tot o listă de întregi, de lungimea primeia, iar al treilea argument este produsul scalar dintre primele două argumente.

Reamintim că:

$$dot(v, w) = \sum_{i=1}^{n} v_i \cdot w_i$$

```
?- dot([2,5,6],[3,4,1], Result).
Result = 32
```

2.9 Exerciţiul 9

Scrieţi un predicat `max/2` care caută elementul maxim dintr-o listă de numere naturale.

```
?- max([4,2,6,8,1], Result).
Result = 8
```

2.10 Exerciţiul 10

Să se definească un predicat `concat_lists/3` care să returneze în al treilea argument concatenarea listelor din primele două argumente.

```
?- concat_lists([1,2,3], [d,e,f,g], X).
X = [1,2,3,d,e,f,g]
```

2.11 Exerciţiul 11

Definiţi un predicat `remove_duplicates/2` care şterge toate duplicatele din lista dată ca prim argument şi întoarce rezultatul în al doilea argument.

```
?- remove_duplicates([a,b,a,c,d,d], List).
List = [b,a,c,d]
```

2.12 Exerciţiul 12

În Prolog avem predefinit predicatul `bagof/3`, cu următoarele argumente:

```
?- bagof(Template, Goal, List).
```

care are ca scop returnarea unei liste (pe ultima poziţie) care conţine toate acele elemente cu aceeaşi structură ca `Template` care satisfac `Goal`. De exemplu, dacă ne amintim baza de cunoştinţe din laboratorul anterior, putem vedea copiii lui `sam` apelând:

```
?- bagof(X, parent_of(X, sam), List).
List = [sandra, ben]
```

(returnează în `List` toţi acei `X` cu proprietatea că `parent_of(X, sam)` este satisfăcut).