

Listas, Recursão e Aleatoriedade

João Davi Rigo Mazzarolo e Luís Henrique Chesani



Listas

- Finitas
- Podem ser heterogêneas
- Exemplo: [a, 1, cachorro, X].

Alguns predicados

```
?- member(4, [7, 5, 4, 3]).  
true ;
```

```
?- member(X, [7, 3, 1]).  
X = 7 ;  
X = 3 ;  
X = 1.
```

```
?- length([3, 4, 5, 7], 5).  
false.
```

```
?- length([3, 4, 5, 7], X).  
X = 4.
```

Alguns predicados

```
?- nth0(3, [2, 7, 8, cachorro, jeferson], X).  
X = cachorro.
```

```
?- nth1(3, [2, 7, 8, cachorro, jeferson], X).  
X = 8.
```

```
?- last([a,b,c,d],X).  
X = d
```

```
?- last([a,b,c,d],d).  
true
```

Alguns predicados

```
?- reverse([a,b,c,d],L).  
L = [d, c, b, a]
```

```
?- append([a,b],[c,d],L).  
L = [a, b, c, d].
```

```
?- reverse([a,b,c,d],[d, c, b, a]).  
true
```

```
?- append([a,b],[c,d],[a, b, c, d]).  
true
```

Aleatoriedade

```
?- random(1, 5, X).  
X = 4.
```


```
?- random(1, 5, X).  
X = 1.
```

```
?- random(1, 5, X).  
X = 2.
```

X vai receber um número aleatório entre [1, 5).

Recursividade

```
sucessor(X, Z) :- antecessor(X, Z).  
sucessor(X, Z) :- antecessor(X, Y),sucessor(Y, Z).  
  
antecessor(dogson, cachorro).  
antecessor(cachorro, joao).
```

```
?- sucessor(dogson, X).  
X = cachorro ;  
X = joao ;  
false.  
  
?- 
```

Recursividad

```
pares_ordenados([], [], []).  
pares_ordenados([X | XS], [Y | YS], [Z | ZS]) :-  
    Z = [X, Y],  
    pares_ordenados(XS, YS, ZS).
```

```
?- pares_ordenados([1, 7, 2, 9], [a, z, c, y], X).  
X = [[1, a], [7, z], [2, c], [9, y]].
```



```
1  contato(oscar, dogson).
2  contato(dogson, cachorro).
3  contato(cachorro, sasuke).
4  contato(sasuke, null).
5
6  status(oscar, 100). %nativo
7  status(dogson, 0).
8  status(sasuke, 0).
9  status(cachorro, 0).
10
11 fluxo(null, []). %gera fluxo da doenca com base nos contatos contatos
12 fluxo(Nome, [X | XS]) :-
13     contato(Nome, Nome2),
14     fluxo(Nome2, XS),
15     X = Nome.
```

```
17 prob(null, _, []). %gera a lista de probabilidades
18 prob(Contaminado, Prob, [X | XS]) :-
19     status(Contaminado, Coef),
20     contato(Contaminado, Vitima),
21     random(0, Prob, C),
22     X is C + Coef,
23     prob(Vitima, X, XS).
24
25 pares_ordenados([], [], []).
26 pares_ordenados([X | XS], [Y | YS], [Z | ZS]) :-
27     Z = [X, Y],
28     pares_ordenados(XS, YS, ZS).
29
30 status_doenca(null, []).
31 status_doenca(Contaminado, Lista) :-
32     fluxo(Contaminado, X),
33     prob(Contaminado, 1, Y),
34     pares_ordenados(X, Y, Lista).
```

Exercício

Crie um arquivo .pl que respeite os seguintes itens

1. Crie um predicado que retorne uma lista de hierarquia alimentar com as seguintes regras:
 2. Deve ser possível consultar quais animais (ou sangue) estão sendo digeridos por determinado animal.
- O predicado recebe o animal topo da cadeia alimentar.
 - O predicado utiliza recursão.
 - A hierarquia utiliza recursão.
 - O mosquito come o sangue.
 - O sapo come o mosquito.
 - A cobra come o sapo.
-
- Pode existir uma relação de “comeu(sangue, null)” para simplificar a resolução.