Universidade Regional de Blumenau
Centro de Ciências Exatas e Naturais
Departamento de Sistemas e Computação

**Lógica para Ciências da Computação**
Professor: Jomi Fred Hübner

# Exercício em Laboratório: **Prolog – Listas**

## Sumário



## 1 Objetivo

Aprender a manipular listas no Prolog.

## 2 Predicados Prontos

Uma lista em prolog é uma sequência de termos separados por vírgula e delimitados por colchetes, por exemplo:

- [a,b,c]
- [] (lista vazia)
- [1,2,3]
- [a,1,'casa cheia',homem(bill)]

Os principais predicados e seus argumentos para manipulação de listas são os seguintes:

- member(?Elemento, ?Lista): é verdade se ?Elemento é um dos elementos da ?Lista. A ? antes do argumento indica que ele pode ser um valor ou uma variável.

  Teste os seguintes exemplos:

  - ?- member(b,[a,b,c]).
  - ?- member(X,[a,b,c]).

- append(?Lista1, ?Lista2, ?Lista3): é verdade se a ?Lista3 é a junção dos elementos da ?Lista1 com a ?Lista2

  Teste os seguintes exemplos:

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

- delete(+Lista1, +Elemento, ?Lista2): é verdade se a ?Lista2 é a ?lista1 sem o ?Elemento. O + na frente de um argumento indica que deve ser passado um valor (e não uma variável) para o predicado.

  Teste os seguintes exemplos:

  - ?- delete([a,b,c],b,L).
  - ?- delete([a,b,b,c],b,L).

- length(?Lista, ?Inteiro): é verdade se o número de elementos da ?Lista é igual ao ?Inteiro.

  Teste os seguintes exemplos:

  - ?- length([a,b,c],2).
  - ?- length([a,b,b,c],Tam).

- nth1(?Inteiro, ?Lista, ?Elemento): é verdade se ?Elemento está na posição ?Inteiro da ?Lista (considerando a primeira posição como sendo 1).

  Teste os seguintes exemplos:

  - ?- nth1(2,[a,b,c],X).
  - ?- nth1(X,[a,b,b,c],b).

  (Repare que um mesmo predicados pode ser utilizado para mais de uma função.)

- sort(+Lista1, -Lista2): é verdade se ?Lista2 tem os elementos da ?Lista1 ordenados. O - na frente de um argumento indica que deve ser passado uma variável (e não um valor) para o predicado.

  Teste os seguintes exemplos:

  - ?- sort([c,z,a,b],X).
  - ?- sort([a,b,b,c],X).

- findall(-X, +Meta, -Lista2): cria uma lista com todos os valores de X que satisfazem a Meta.

  Carregue o programa da genealogia, e teste os seguintes exemplos:

  - ?- findall(X,genitor(X,bob),L).
  - ?- findall(X,irmao(X,Y),L).
  - ?- findall(X,ant(X,jim),L).

No help do SWI-Prolog podem ser obtidas mais informações e predicados prontos (Built-in predicates). Para ativar o help, digite

```
?- help.
```

### 2.1 Exercícios

Monte consultas no prolog para

1. Criar um lista com todos os genitores.
2. Criar um lista com todos os genitores, mas o mesmo genitor não deve aparecer duas vezes na lista.
3. Da lista de todos os genitores, imprimir somente o último da lista (utilize o comando write para imprimir).
4. Criar uma lista com todos os antecessores de Jim e Ana.

## 3 Unificação com listas

O operador de unificação (o =) permite obter vários tipos de informação de uma lista.

Teste as seguintes consultas:

- [a,b,c] = L. (L unifica com toda a lista.)
- [a,b,c] = [X1,X2,X3]. (cada variável unifica com cada elemento da lista.)
- [a,b,c] = [C|R]. (C unifica com a cabeça da lista, o "a", e R unifica com o restante da lista, a lista "[b,c]". O caracter "|" distingue a cabeça da lista do restante.)

### 3.1 Exercícios

Qual o resultados das seguintes unificações (tente prever o resultado antes de executar o comando):

1. [a,b,c,d,e,f] = [C|R].
2. [a,b,c,d,e,f] = [C,R].
3. [a,b] = [C|R].

4. [a,b] = [C,R].
5. [a,b,c] = [C|R].
6. [a,b,c] = [C1,C2|R].
7. [a,b,c] = [C1,C2,C3|R].
8. [a,b,c] = [C1,C2,C3,C4|R].
9. [a] = [C].
10. [a] = [C|R].
11. [a] = [C1,C2|R].

## 4 Regras utilizando listas

A definição de regras para processar listas normalmente adota uma abordagem recursiva e, portanto, uma regras específica para término da recursão e uma regra geral. Por exemplo, o predicado `tamanho` poderia ser assim definido:

```
% o tamanho de uma lista vazia é 0
tamanho([],0).

% o tamanho de uma lista que tem uma
% cabeça é o tamanho do resto + 1
tamanho([C|R],T) :- tamanho(R,TR), T is TR + 1.
```

Obs.: "is" é o comando de avaliação de expressões aritméticas do prolog.

Um predicado para imprimir toda uma lista é

```
% para uma lista vazia, não se imprime nada
imprime([]).

% para uma lista que tem uma cabeça, imprime-se
% a cabeça e depois o resto
imprime([C|R]) :- write(C), nl, imprime(R).
```

Em uma regra, para incluir um elemento em uma lista faz-se uma unificação com a cabeça da lista. Por exemplo, um predicado que, a partir de uma lista de pessoas, cria uma lista com todas que são mulheres:

```
% em uma lista vazia não há mulheres
mulheres([],[]).

% em uma lista que tem uma cabeça, se a cabeça for
% mulher, é incluída na lista e se continua
% verificando os demais elementos da lista
mulheres([C|RGeral],[C|RMulheres]) :-
  mulher(C),
  mulheres(RGeral, RMulheres).
```

```
% em uma lista que tem uma cabeça, se a cabeça NÃO for
% mulher, continua-se verificando o restante da lista
mulheres([C|RGeral],RMulheres) :-
  not(mulher(C)),
  mulheres(RGeral, RMulheres).
```

### 4.1 Exercícios

- Verifique na literatura recomendada no final deste documento como são definidos os predicados prontos do prolog (member, append, etc.).
- Defina um predicado que recebe três listas como argumento. Tal predicado deve ser verdade quando na terceira lista constar apenas elementos comuns entre as duas primeiras lista. Exemplo de uso:

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

  L = [b,c]

Yes
?-
```

- Usando o predicado acima, faça uma consulta que recupere os antecedentes comuns de Jim e Ana.
- No exemplo da genealogia, inclua um predicado `idade` com dois argumentos: o nome da pessoa e sua idade. Por exemplo:

```
idade(bob,50).
idade(ana,23).
```

  Defina um predicado que recebe uma lista de pessoas como parâmetro e imprime somente aquelas que tem mais de 21 anos.

## 5 Bibliografia


- BRATKO, Ivan. **Prolog programming for artificial intelligence**. 2.ed. Wokingham : Addison-Wesley, 1990.
- STERLING, Leon; SHAPIRO, Ehud. **The art of Prolog**: advanced programming techniques. 2.ed. Cambridge : MIT, 1994. (p.411-478)
- CASANOVA, Marco Antonio, et al. **Programação em lógica e a linguagem PROLOG**. São Paulo : E. Blucher, c1987.