GSI010 - Programação Lógica
Listas

Lista

Definição

Uma lista é uma sequência finita de elementos.

Exemplos

```
1 ?- Lista = [brasil, uruguai, argentina, paraguai].
2 Lista = [brasil, uruguai, argentina, paraguai].
```

Lista

Definição

Uma lista é uma sequência finita de elementos.

Exemplos

```
1 ?- Lista = [brasil, uruguai, argentina, paraguai].
2 Lista = [brasil, uruguai, argentina, paraguai].
3
4 ?- ListaDeFatos = [idade(32), idade(21), idade(45)].
```

Lista

Definição

Uma lista é uma sequência finita de elementos.

Exemplos

```
1 ?- Lista = [brasil, uruguai, argentina, paraguai].
2 Lista = [brasil, uruguai, argentina, paraguai].
3
4 ?- ListaDeFatos = [idade(32), idade(21), idade(45)].
5
6 ?- ListaInfo = [ruaNaves, nome(maria), idade(20), 50,
      12-05-9a, 1109.01].
7
8 ?- [ruaNaves, nome(maria)] = [X, nome(Y)].
```

Lista

Definição

Uma lista é uma sequência finita de elementos.

Exemplos

```
1  ?- Lista = [brasil, uruguai, argentina, paraguai].
2  Lista = [brasil, uruguai, argentina, paraguai].
3
4  ?- ListaDeFatos = [idade(32), idade(21), idade(45)].
5
6  ?- ListaInfo = [ruaNaves, nome(maria), idade(20), 50,
       12-05-9a, 1109.01].
7
8  ?- [ruaNaves, nome(maria)] = [X, nome(Y)].
9  X = ruaNaves,
10 Y = maria.
```

Lista

Partes

Listas não vazias têm um elemento chamado cabeça e uma lista de elementos chamada cauda.

Decomposição

Operador decomposição, uma barra vertical, | separa cabeça e cauda de uma lista.

Operador decomposição “barra vertical”

```
?- [Cabeca | Cauda] = [brasil, uruguai, argentina,
    paraguai].
```

Operador decomposição “barra vertical”

```
1 ?- [Cabeca | Cauda] = [brasil, uruguai, argentina,
      paraguai].
2 Cabeca = brasil,
3 Cauda = [uruguai, argentina, paraguai].
```

Operador decomposição “barra vertical”

```
1 ?- [Cabeca | Cauda] = [brasil, uruguai, argentina,
      paraguai].
2 Cabeca = brasil,
3 Cauda = [uruguai, argentina, paraguai].
4
5 ?- [Head|Tail] = [paraguai].
```

Operador decomposição “barra vertical”

```
1 ?- [Cabeca | Cauda] = [brasil, uruguai, argentina,
      paraguai].
2 Cabeca = brasil,
3 Cauda = [uruguai, argentina, paraguai].
4
5 ?- [Head|Tail] = [paraguai].
6 Head = paraguai,
7 Tail = []. % lista vazia
8
9 ?- [X,Y,Z] = [a,b,c,d].
```

Operador decomposição “barra vertical”

```
1  ?- [Cabeca | Cauda] = [brasil, uruguai, argentina,
       paraguai].
2  Cabeca = brasil,
3  Cauda = [uruguai, argentina, paraguai].
4
5  ?- [Head|Tail] = [paraguai].
6  Head = paraguai,
7  Tail = []. % lista vazia
8
9  ?- [X,Y,Z] = [a,b,c,d].
10 false.
11
12 ?- [H|T]=[].
```

Operador decomposição “barra vertical”

```
1  ?- [Cabeca | Cauda] = [brasil, uruguai, argentina,
       paraguai].
2  Cabeca = brasil,
3  Cauda = [uruguai, argentina, paraguai].
4
5  ?- [Head|Tail] = [paraguai].
6  Head = paraguai,
7  Tail = []. % lista vazia
8
9  ?- [X,Y,Z] = [a,b,c,d].
10 false.
11
12 ?- [H|T]=[].
13 false.
```

Operador decomposição "barra vertical"

```
1  ?- [Cabeca | Cauda] = [brasil, uruguai, argentina,
       paraguai].
2  Cabeca = brasil,
3  Cauda = [uruguai, argentina, paraguai].
4
5  ?- [Head|Tail] = [paraguai].
6  Head = paraguai,
7  Tail = []. % lista vazia
8
9  ?- [X,Y,Z] = [a,b,c,d].
10 false.
11
12 ?- [H|T]=[].
13 false.
```

Lista vazia não tem cabeça nem cauda e costuma ser usada como caso base em recursão.

Cabeça e cauda

```
1 ?- [Cab|Cauda] = [cabeça, ombro, joelho, pé].
```

Cabeça e cauda

```
1 ?- [Cab|Cauda] = [cabeça, ombro, joelho, pé].
2 Cab='cabeça',
3 Cauda=[ombro, joelho, 'pé'].
```

Cabeça e cauda

```
1 ?- [Cab|Cauda] = [cabeça, ombro, joelho, pé].
2 Cab='cabeça',
3 Cauda=[ombro, joelho, 'pé'].
4
5 ?- [Cab|[Cab2|Cauda]] = [cabeça, ombro, joelho, pé].
```

Cabeça e cauda

```
1 ?- [Cab|Cauda] = [cabeça, ombro, joelho, pé].
2 Cab='cabeça',
3 Cauda=[ombro, joelho, 'pé'].
4
5 ?- [Cab|[Cab2|Cauda]] = [cabeça, ombro, joelho, pé].
6 Cab = 'cabeça',
7 Cab2 = ombro,
8 Cauda = [joelho, 'pé'].
```

Cabeça e cauda

```
1  ?- [Cab|Cauda] = [cabeça, ombro, joelho, pé].
2  Cab='cabeça',
3  Cauda=[ombro, joelho, 'pé'].
4
5  ?- [Cab|[Cab2|Cauda]] = [cabeça, ombro, joelho, pé].
6  Cab = 'cabeça',
7  Cab2 = ombro,
8  Cauda = [joelho, 'pé'].
9
10 ?- [Cab,Cab2|Cauda] = [cabeça, ombro, joelho, pé].
```

Cabeça e cauda

```
1  ?- [Cab|Cauda] = [cabeça, ombro, joelho, pé].
2  Cab='cabeça',
3  Cauda=[ombro, joelho, 'pé'].
4
5  ?- [Cab|[Cab2|Cauda]] = [cabeça, ombro, joelho, pé].
6  Cab = 'cabeça',
7  Cab2 = ombro,
8  Cauda = [joelho, 'pé'].
9
10 ?- [Cab,Cab2|Cauda] = [cabeça, ombro, joelho, pé].
11 Cab = 'cabeça',
12 Cab2 = ombro,
13 Cauda = [joelho, 'pé'].
```

Cabeça e cauda

```
1  ?- [Cab|Cauda] = [cabeça, ombro, joelho, pé].
2  Cab='cabeça',
3  Cauda=[ombro, joelho, 'pé'].
4
5  ?- [Cab|[Cab2|Cauda]] = [cabeça, ombro, joelho, pé].
6  Cab = 'cabeça',
7  Cab2 = ombro,
8  Cauda = [joelho, 'pé'].
9
10 ?- [Cab,Cab2|Cauda] = [cabeça, ombro, joelho, pé].
11 Cab = 'cabeça',
12 Cab2 = ombro,
13 Cauda = [joelho, 'pé'].
14
15 ?- [Cab,Cab2|Cauda] = [cabeça, ombro, [joelho, pé]], [
       Cab3|Fim]=Cauda.
```

Cabeça e cauda

```
1  ?- [Cab|Cauda] = [cabeça, ombro, joelho, pé].
2  Cab='cabeça',
3  Cauda=[ombro, joelho, 'pé'].
4
5  ?- [Cab|[Cab2|Cauda]] = [cabeça, ombro, joelho, pé].
6  Cab = 'cabeça',
7  Cab2 = ombro,
8  Cauda = [joelho, 'pé'].
9
10 ?- [Cab,Cab2|Cauda] = [cabeça, ombro, joelho, pé].
11 Cab = 'cabeça',
12 Cab2 = ombro,
13 Cauda = [joelho, 'pé'].
14
15 ?- [Cab,Cab2|Cauda] = [cabeça, ombro, [joelho, pé]], [
       Cab3|Fim]=Cauda.
16 Cab = 'cabeça',
17 Cab2 = ombro,
18 Cauda = [[joelho, 'pé']],
19 Cab3 = [joelho, 'pé'],
20 Fim = [].
```

Operações básicas

Inicialmente podemos

- verificar se elemento pertence a uma lista
- construir uma lista
- remover elemento de uma lista
- concatenar duas listas

constroi/3

Predicado: constroi/3

```
constroi(H,T,[H|T]).
```

```
1 ?- constroi(argentina, brasil, L).
```

constroi/3

Predicado: constroi/3

```
constroi(H,T,[H|T]).
```

```
1 ?- constroi(argentina, brasil, L).
2 L = [argentina|brasil].
```

constroi/3

Predicado: constroi/3

```
constroi(H,T,[H|T]).
```

```
1 ?- constroi(argentina, brasil, L).
2 L = [argentina|brasil].
3
4 ?- constroi(argentina, [brasil, colombia], L).
```

constroi/3

Predicado: constroi/3

```
constroi(H,T,[H|T]).
```

```
1 ?- constroi(argentina, brasil, L).
2 L = [argentina|brasil].
3
4 ?- constroi(argentina, [brasil, colombia], L).
5 L = [argentina, brasil, colombia].
```

constroi/3

Predicado: constroi/3

```
constroi(H,T,[H|T]).
```

```
1 ?- constroi(argentina, brasil, L).
2 L = [argentina|brasil].
3
4 ?- constroi(argentina, [brasil, colombia], L).
5 L = [argentina, brasil, colombia].
6
7 ?- constroi(argentina, [], L).
```

constroi/3

Predicado: constroi/3

```
constroi(H,T,[H|T]).
```

```
1 ?- constroi(argentina, brasil, L).
2 L = [argentina|brasil].
3
4 ?- constroi(argentina, [brasil, colombia], L).
5 L = [argentina, brasil, colombia].
6
7 ?- constroi(argentina, [], L).
8 L = [argentina].
```

constroi/3

Predicado: constroi/3

```
constroi(H,T,[H|T]).
```

```
1  ?- constroi(argentina, brasil, L).
2  L = [argentina|brasil].
3
4  ?- constroi(argentina, [brasil, colombia], L).
5  L = [argentina, brasil, colombia].
6
7  ?- constroi(argentina, [], L).
8  L = [argentina].
9
10 ?- constroi(argentina, Cauda, [argentina, brasil,
       colombia]).
```

constroi/3

Predicado: constroi/3

```
constroi(H,T,[H|T]).
```

```
1  ?- constroi(argentina, brasil, L).
2  L = [argentina|brasil].
3
4  ?- constroi(argentina, [brasil, colombia], L).
5  L = [argentina, brasil, colombia].
6
7  ?- constroi(argentina, [], L).
8  L = [argentina].
9
10 ?- constroi(argentina, Cauda, [argentina, brasil,
       colombia]).
11 Cauda = [brasil, colombia].
```

constroi/3

Predicado: constroi/3

```
constroi(H,T,[H|T]).
```

```
1  ?- constroi(argentina, brasil, L).
2  L = [argentina|brasil].
3
4  ?- constroi(argentina, [brasil, colombia], L).
5  L = [argentina, brasil, colombia].
6
7  ?- constroi(argentina, [], L).
8  L = [argentina].
9
10 ?- constroi(argentina, Cauda, [argentina, brasil,
       colombia]).
11 Cauda = [brasil, colombia].
12
13 ?- constroi([argentina, brasil], [colombia, dinamarca],
       L).
```

constroi/3

Predicado: constroi/3

```
constroi(H,T,[H|T]).
```

```
1  ?- constroi(argentina, brasil, L).
2  L = [argentina|brasil].
3
4  ?- constroi(argentina, [brasil, colombia], L).
5  L = [argentina, brasil, colombia].
6
7  ?- constroi(argentina, [], L).
8  L = [argentina].
9
10 ?- constroi(argentina, Cauda, [argentina, brasil,
       colombia]).
11 Cauda = [brasil, colombia].
12
13 ?- constroi([argentina, brasil], [colombia, dinamarca],
       L).
14 L = [[argentina, brasil], colombia, dinamarca].
```

constroi/3

Predicado: constroi/3

`constroi(H,T,[H|T]).`

```
1  ?- constroi(argentina, brasil, L).
2  L = [argentina|brasil].
3
4  ?- constroi(argentina, [brasil, colombia], L).
5  L = [argentina, brasil, colombia].
6
7  ?- constroi(argentina, [], L).
8  L = [argentina].
9
10 ?- constroi(argentina, Cauda, [argentina, brasil,
       colombia]).
11 Cauda = [brasil, colombia].
12
13 ?- constroi([argentina, brasil], [colombia, dinamarca],
       L).
14 L = [[argentina, brasil], colombia, dinamarca].
15
16 ?- constroi(argentina, C, [brasil, colombia]).
```

constroi/3

Predicado: constroi/3

```
constroi(H,T,[H|T]).
```

```
?- constroi(argentina, brasil, L).
L = [argentina|brasil].

?- constroi(argentina, [brasil, colombia], L).
L = [argentina, brasil, colombia].

?- constroi(argentina, [], L).
L = [argentina].

?- constroi(argentina, Cauda, [argentina, brasil,
    colombia]).
Cauda = [brasil, colombia].

?- constroi([argentina, brasil], [colombia, dinamarca],
     L).
L = [[argentina, brasil], colombia, dinamarca].

?- constroi(argentina, C, [brasil, colombia]).
false.
```

membro/2

Predicado: membro/2

Verifica de X é elemento membro de L: `membro(X, L).`

Definição

```
1 membro(X, [X|C]).
2 membro(X, [H|C]):- membro(X, C).
```

Exemplos: `membro/2`

```
1 ?- membro(x, [y,w,z,x]).
```

Exemplos: membro/2

```
1 ?- membro(x, [y,w,z,x]).
2 true ;
```

Exemplos: membro/2

```
1 ?- membro(x, [y,w,z,x]).
2 true ;
3 false.
```

Exemplos: membro/2

```
1 ?- membro(x, [y,w,z,x]).
2 true ;
3 false.
4
5 ?- membro(x,[y,[w,z,x]]).
```

Exemplos: membro/2

```
1 ?- membro(x, [y,w,z,x]).
2 true ;
3 false.
4
5 ?- membro(x,[y,[w,z,x]]).
6 false.
```

Exemplos: membro/2

```
1 ?- membro(x, [y,w,z,x]).
2 true ;
3 false.
4
5 ?- membro(x,[y,[w,z,x]]).
6 false.
7
8 ?- membro([w,z,x], [y,[w,z,x]]).
```

Exemplos: membro/2

```
1 ?- membro(x, [y,w,z,x]).
2 true ;
3 false.
4
5 ?- membro(x,[y,[w,z,x]]).
6 false.
7
8 ?- membro([w,z,x], [y,[w,z,x]]).
9 true ;
```

Exemplos: membro/2

```
1  ?- membro(x, [y,w,z,x]).
2  true ;
3  false.
4
5  ?- membro(x,[y,[w,z,x]]).
6  false.
7
8  ?- membro([w,z,x], [y,[w,z,x]]).
9  true ;
10 false.
```

Exemplos: membro/2

```
1  ?- membro(x, [y,w,z,x]).
2  true ;
3  false.
4
5  ?- membro(x,[y,[w,z,x]]).
6  false.
7
8  ?- membro([w,z,x], [y,[w,z,x]]).
9  true ;
10 false.
11
12 ?- membro(X, [y,[w,z,x]]).
```

Exemplos: membro/2

```
1  ?- membro(x, [y,w,z,x]).
2  true ;
3  false.
4
5  ?- membro(x,[y,[w,z,x]]).
6  false.
7
8  ?- membro([w,z,x], [y,[w,z,x]]).
9  true ;
10 false.
11
12 ?- membro(X, [y,[w,z,x]]).
13 X = y ;
```

Exemplos: membro/2

```
1  ?- membro(x, [y,w,z,x]).
2  true ;
3  false.
4
5  ?- membro(x,[y,[w,z,x]]).
6  false.
7
8  ?- membro([w,z,x], [y,[w,z,x]]).
9  true ;
10 false.
11
12 ?- membro(X, [y,[w,z,x]]).
13 X = y ;
14 X = [w, z, x] ;
```

Exemplos: membro/2

```
1  ?- membro(x, [y,w,z,x]).
2  true ;
3  false.
4
5  ?- membro(x,[y,[w,z,x]]).
6  false.
7
8  ?- membro([w,z,x], [y,[w,z,x]]).
9  true ;
10 false.
11
12 ?- membro(X, [y,[w,z,x]]).
13 X = y ;
14 X = [w, z, x] ;
15 false.
```

Exemplos: membro/2

```
1  ?- membro(x, [y,w,z,x]).
2  true ;
3  false.
4
5  ?- membro(x,[y,[w,z,x]]).
6  false.
7
8  ?- membro([w,z,x], [y,[w,z,x]]).
9  true ;
10 false.
11
12 ?- membro(X, [y,[w,z,x]]).
13 X = y ;
14 X = [w, z, x] ;
15 false.
16
17 ?- membro([X,Y,Z], [y,[w,z,x]]).
```

Exemplos: membro/2

```
1  ?- membro(x, [y,w,z,x]).
2  true ;
3  false.
4
5  ?- membro(x,[y,[w,z,x]]).
6  false.
7
8  ?- membro([w,z,x], [y,[w,z,x]]).
9  true ;
10 false.
11
12 ?- membro(X, [y,[w,z,x]]).
13 X = y ;
14 X = [w, z, x] ;
15 false.
16
17 ?- membro([X,Y,Z], [y,[w,z,x]]).
18 X = w,
```

Exemplos: membro/2

```
1  ?- membro(x, [y,w,z,x]).
2  true ;
3  false.
4
5  ?- membro(x,[y,[w,z,x]]).
6  false.
7
8  ?- membro([w,z,x], [y,[w,z,x]]).
9  true ;
10 false.
11
12 ?- membro(X, [y,[w,z,x]]).
13 X = y ;
14 X = [w, z, x] ;
15 false.
16
17 ?- membro([X,Y,Z], [y,[w,z,x]]).
18 X = w,
19 Y = z,
```

Exemplos: membro/2

```
1  ?- membro(x, [y,w,z,x]).
2  true ;
3  false.
4
5  ?- membro(x,[y,[w,z,x]]).
6  false.
7
8  ?- membro([w,z,x], [y,[w,z,x]]).
9  true ;
10 false.
11
12 ?- membro(X, [y,[w,z,x]]).
13 X = y ;
14 X = [w, z, x] ;
15 false.
16
17 ?- membro([X,Y,Z], [y,[w,z,x]]).
18 X = w,
19 Y = z,
20 Z = x ;
```

Exemplos: membro/2

```
1  ?- membro(x, [y,w,z,x]).
2  true ;
3  false.
4
5  ?- membro(x,[y,[w,z,x]]).
6  false.
7
8  ?- membro([w,z,x], [y,[w,z,x]]).
9  true ;
10 false.
11
12 ?- membro(X, [y,[w,z,x]]).
13 X = y ;
14 X = [w, z, x] ;
15 false.
16
17 ?- membro([X,Y,Z], [y,[w,z,x]]).
18 X = w,
19 Y = z,
20 Z = x ;
21 false.
```

Exemplos: membro/2

```
1  ?- membro(x, [y,w,z,x]).
2  true ;
3  false.
4
5  ?- membro(x,[y,[w,z,x]]).
6  false.
7
8  ?- membro([w,z,x], [y,[w,z,x]]).
9  true ;
10 false.
11
12 ?- membro(X, [y,[w,z,x]]).
13 X = y ;
14 X = [w, z, x] ;
15 false.
16
17 ?- membro([X,Y,Z], [y,[w,z,x]]).
18 X = w,
19 Y = z,
20 Z = x ;
21 false.
```

Operação: concatenar/3

Predicado: concatenar/3

Dadas duas listas L1 e L2, obter uma terceira lista com o conteúdo de L1 seguido de L2.

```
1  concatenar([],L,L).
2  concatenar([H|T], L, [H|TL]) :- concatenar(T,L,
       TL).
```

Exemplos de concatenar/3

```
1 ?- concatenar([a,b,c],[d,e,f], L).
```

Exemplos de concatenar/3

```
1 ?- concatenar([a,b,c],[d,e,f], L).
2 L = [a, b, c, d, e, f].
```

Exemplos de concatenar/3

```
1 ?- concatenar([a,b,c],[d,e,f], L).
2 L = [a, b, c, d, e, f].
3
4 ?- concatenar([a,[b,c],h],[d,e,f], L).
```

Exemplos de concatenar/3

```
1 ?- concatenar([a,b,c],[d,e,f], L).
2 L = [a, b, c, d, e, f].
3
4 ?- concatenar([a,[b,c],h],[d,e,f], L).
5 L = [a, [b, c], h, d, e, f].
```

Exemplos de concatenar/3

```
1 ?- concatenar([a,b,c],[d,e,f], L).
2 L = [a, b, c, d, e, f].
3
4 ?- concatenar([a,[b,c],h],[d,e,f], L).
5 L = [a, [b, c], h, d, e, f].
6
7 ?- concatenar([a], X, [a,[b,c],h]).
```

Exemplos de concatenar/3

```
1 ?- concatenar([a,b,c],[d,e,f], L).
2 L = [a, b, c, d, e, f].
3
4 ?- concatenar([a,[b,c],h],[d,e,f], L).
5 L = [a, [b, c], h, d, e, f].
6
7 ?- concatenar([a], X, [a,[b,c],h]).
8 X = [[b, c], h].
```

Exemplos de concatenar/3

```
1  ?- concatenar([a,b,c],[d,e,f], L).
2  L = [a, b, c, d, e, f].
3
4  ?- concatenar([a,[b,c],h],[d,e,f], L).
5  L = [a, [b, c], h, d, e, f].
6
7  ?- concatenar([a], X, [a,[b,c],h]).
8  X = [[b, c], h].
9
10 ?- concatenar(L1, L2, [a,[b,c],h]).
```

Exemplos de concatenar/3

```
?- concatenar([a,b,c],[d,e,f], L).
L = [a, b, c, d, e, f].

?- concatenar([a,[b,c],h],[d,e,f], L).
L = [a, [b, c], h, d, e, f].

?- concatenar([a], X, [a,[b,c],h]).
X = [[b, c], h].

?- concatenar(L1, L2, [a,[b,c],h]).
L1 = [],
L2 = [a, [b, c], h] ;
L1 = [a],
L2 = [[b, c], h] ;
L1 = [a, [b, c]],
L2 = [h] ;
L1 = [a, [b, c], h],
L2 = [] ;

?- M=[jan,fev,mar,abr,mai,jun,jul,ago,set,out,nov,dez],
concatenar(Antes, [mai | Depois], M).
```

Exemplos de concatenar/3

```
?- concatenar([a,b,c],[d,e,f], L).
L = [a, b, c, d, e, f].

?- concatenar([a,[b,c],h],[d,e,f], L).
L = [a, [b, c], h, d, e, f].

?- concatenar([a], X, [a,[b,c],h]).
X = [[b, c], h].

?- concatenar(L1, L2, [a,[b,c],h]).
L1 = [],
L2 = [a, [b, c], h] ;
L1 = [a],
L2 = [[b, c], h] ;
L1 = [a, [b, c]],
L2 = [h] ;
L1 = [a, [b, c], h],
L2 = [] ;

?- M=[jan,fev,mar,abr,mai,jun,jul,ago,set,out,nov,dez],
concatenar(Antes, [mai | Depois], M).
Antes = [jan, fev, mar, abr],
Depois = [jun, jul, ago, set, out, nov, dez] ;
false.
```

Operação: `remover/3`

Predicado: `remover/3`

`remover(X, L, L1)` deve remover X de L, originando L1.

Dada lista `[H|C]`

- Se X unifica com H, L1 é a cauda C
- Se X não unifica com H, remover X da cauda e incluir H em L

Operação: `remover/3`

Predicado: `remover/3`

`remover(X, L, L1)` deve remover X de L, originando L1.

Dada lista `[H|C]`

- Se X unifica com H, L1 é a cauda C
- Se X não unifica com H, remover X da cauda e incluir H em L

Exercício: escrever `remover/3`.

Referências

- Luis, A. M. Palazzo, Introdução à programação prolog, Educat, 1997
- Slides profs. Elaine Faria, Hiran Nonato e Gabriel Coutinho - UFU
- Slides da Profa. Solange - ICMC - USP