Aula 4: Listas

- Teoria
  - Introduzir listas, uma estrutura de dados recursiva importante muito usada em programação Prolog.
  - Definir o predicado membro/2, uma ferramenta fundamental do Prolog para a manipulação de listas
  - Ilustar a ideia de recursão sobre listas

membro/2

```
membro(X,[X|T]).
membro(X,[H|T]):- membro(X,T).
```

```
?-
```

membro/2

```
membro(X,[X|T]).
membro(X,[H|T]):- membro(X,T).
```

```
?- membro(iolanda,[iolanda,patricia,vicente,julio]).
```

membro/2

```
membro(X,[X|T]).
membro(X,[H|T]):- membro(X,T).
```

```
?- membro(iolanda,[iolanda,patricia,vicente,julio]).
true
```

Listas

- Uma lista é uma sequência finita de elementos
- Exemplos de listas em Prolog:

  [maria, vicente, julio, iolanda]

  [maria, ladrao(mel), X, 2, maria]

  [ ]

  [maria, [vicente, julio], [beto, amigo(beto)]]

  [[ ], morto(z), [2, [b,c]], [ ], Z, [2, [b,c]]]

Coisas importantes sobre listas

- Os elementos de uma lista ficam entre colchetes.
- O comprimento de uma lista é o número de elementos que ela possui.
- ***Qualquer*** tipo de termos em Prolog podem ser elementos de uma lista.
- Existe uma lista especial:
  a lista vazia [ ]

Cabeça e cauda

- Uma lista não vazia consiste de duas partes:
  - A cabeça
  - A cauda
- A cabeça é o primeiro item da lista
- A cauda é todo o resto
  - A cauda é a lista que sobra quando retiramos o primeiro elemento.
  - A cauda de uma lista sempre é uma lista.

Cabeça e cauda - Exemplo 1

- [maria, vicente, julio, iolanda]

  Cabeça:
  Cauda:

Cabeça e cauda - Exemplo 1

- [maria, vicente, julio, iolanda]

  Cabeça: maria
  Cauda:

Cabeça e cauda - Exemplo 1

- [maria, vicente, julio, iolanda]

  Cabeça: maria
  Cauda: [vicente, julio, iolanda]

Cabeça e cauda - Exemplo 2

- [[ ], morto(z), [2, [b,c]], [ ], Z, [2, [b,c]]]

  Cabeça:
  Cauda:

Cabeça e cauda - Exemplo 2

- [[ ], morto(z), [2, [b,c]], [ ], Z, [2, [b,c]]]

  Cabeça: [ ]
  Cauda:

Cabeça e cauda - Exemplo 2

- [[ ], morto(z), [2, [b,c]], [ ], Z, [2, [b,c]]]

  Cabeça: [ ]
  Cauda: [morto(z), [2, [b,c]], [ ], Z, [2, [b,c]]]

Cabeça e cauda - Exemplo 3

- [morto(z)]

  Cabeça:
  Cauda:

Cabeça e cauda - Exemplo 3

- [morto(z)]

  Cabeça: morto(z)
  Cauda:

Cabeça e cauda - Exemplo 3

- [morto(z)]

  Cabeça: morto(z)
  Cauda: []

Cabeça e cauda da lista vazia

- A lista vazia não possui cabeça nem cauda
- Para o Prolog, [ ] é uma lista simples especial sem qualquer estrutura interna
- A lista vazia desempenha um importante papel nos predicados recursivos para processamento de listas em Prolog

O operador |

- Prolog possui um operador especial | que pode ser usado para decompor uma lista em sua cabeça e cauda.
- O operador | é uma peça chave na escrita em Prolog de predicados para a manipulação de listas.

O operador |

```
?- [Cab|Cauda] = [maria, vicent, julio, iolanda].
```

O operador |

```
?- [Cab|Cauda] = [maria, vicent, julio, iolanda].

Cab = maria
Cauda = [vicent, julio, iolanda]


?-
```

O operador |

```
?- [X|Y] = [maria, vicente, julio, iolanda].
```

O operador |

```
?- [X|Y] = [maria, vicente, julio, iolanda].

X = maria
Y = [vicent,julio,iolanda]

?-
```

O operador |

```
?- [X|Y] = [ ].
```

O operador |

```
?- [X|Y] = [ ].
false
```

O operador |

```
?- [X,Y|Cauda] = [[ ], morto(z), [2, [b,c]], [], Z, [2, [b,c]]] .
```

O operador |

```
?- [X,Y|Cauda] = [[ ], morto(z), [2, [b,c]], [], Z, [2, [b,c]]] .

X = [ ]
```

O operador |

```
?- [X,Y|Cauda] = [[ ], morto(z), [2, [b,c]], [], Z, [2, [b,c]]] .

X = [ ]
Y = morto(z)
```

O operador |

```
?- [X,Y|Cauda] = [[ ], morto(z), [2, [b,c]], [], Z, [2, [b,c]]] .

X = [ ]
Y = morto(z)
Z = _G4543
```

O operador |

```
?- [X,Y|Cauda] = [[ ], morto(z), [2, [b,c]], [], Z, [2, [b,c]]] .

X = [ ]
Y = morto(z)
Z = _G4543
Cauda = [[2, [b,c]], [ ], Z, [2, [b,c]]]
```

O operador |

```
?- [X|Y] = [maria, vicente, julio, iolanda].
```

O operador |

```
?- [X|Y] = [maria, vicente, julio, iolanda].

X = maria
Y = [vicent,julio,iolanda]


?-
```

Variável anônima

- Assuma que estejamos interessados no segundo e quarto elementos de uma lista

```
?- [X1,X2,X3,X4|Cauda] = [maria, vicente, marcelo, jonas, iolanda].
X1 = maria
X2 = vicente
X3 = marcelo
X4 = jonas
Cauda = [iolanda]

?-
```

Variáveis anônimas

- Existe uma forma mais simples de obter somente a informação que desejamos:

```
?- [ _,X2, _,X4|_ ] = [maria, vicente, marcelo, jonas, iolanda].
X2 = vicente
X4 = jonas

?-
```

- O sublinhado é a variável anônima

A variável anônima

- É usada quando se necessita utilizar uma variável, mas não se está interessado no valor que o Prolog instanciou para ela.
- Cada ocorrência da variável anônima é independente, i.e., pode ser ligada a algo diferente.

Membro

- Um dos fatos mais básicos que gostaríamos de saber é se alguma coisa é ou não elemento de uma lista.
- Assim, vamos escrever um predicado que ao receber uma termo X e uma lista L, responderá se X pertence ou não a L.
- Este predicado é normalmente chamado membro/2

membro/2

```
membro(X,[X|T]).
membro(X,[H|T]):- membro(X,T).
```

```
?-
```

membro/2

```
membro(X,[X|T]).
membro(X,[H|T]):- membro(X,T).
```

```
?- membro(iolanda,[iolanda,patricia,vicente,julio]).
```

membro/2

```
membro(X,[X|T]).
membro(X,[H|T]):- membro(X,T).
```

```
?- membro(iolanda,[iolanda,patricia,vicente,julio]).
true
```

membro/2

```
membro(X,[X|T]).
membro(X,[H|T]):- membro(X,T).
```

```
?- membro(vicente,[iolanda,patricia,vicente,julio]).
```

membro/2

```
membro(X,[X|T]).
membro(X,[H|T]):- membro(X,T).
```

```
?- membro(vicente,[iolanda,patricia,vicente,julio]).
true
```

membro/2

```
membro(X,[X|T]).
membro(X,[H|T]):- membro(X,T).
```

```
?- membro(zeca,[iolanda,patricia,vicente,julio]).
```

membro/2

```
membro(X,[X|T]).
membro(X,[H|T]):- membro(X,T).
```

```
?- membro(zeca,[iolanda,patricia,vicente,julio]).
false
```

membro/2

```
membro(X,[X|T]).
membro(X,[H|T]):- membro(X,T).
```

```
?- membro(X,[iolanda,patricia,vicente,julio]).
```

membro/2

```
membro(X,[X|T]).
membro(X,[H|T]):- membro(X,T).
```

```
?- membro(X,[iolanda,patricia,vicente,julio]).
X = iolanda;
```

membro/2

```
membro(X,[X|T]).
membro(X,[H|T]):- membro(X,T).
```

```
?- membro(X,[iolanda,patricia,vicente,julio]).
X = iolanda;
X = patricia;
```

membro/2

```
membro(X,[X|T]).
membro(X,[H|T]):- membro(X,T).
```

```
?- membro(X,[iolanda,patricia,vicente,julio]).
X = iolanda;
X = patricia;
X = vicente;
```

membro/2

```
membro(X,[X|T]).
membro(X,[H|T]):- membro(X,T).
```

```
?- membro(X,[iolanda,patricia,vicente,julio]).
X = iolanda;
X = patricia;
X = vicente;
X = julio.
```

Reescrevendo membro/2

```
membro(X,[X|_]).
membro(X,[_|T]):- membro(X,T).
```

Recursão em listas

- O predicado membro/2 percorre recursivamente uma lista
  - Faça alguma coisa com a cabeça, e depois
  - Recursivamente faça a mesma coisa com a cauda.
- Esta técnica é muito comum em Prolog e, portanto, é muito importante que você a domine.
- Assim, vamos observar um outro exemplo!

Exemplo: a2b/2

- O predicado a2b/2 recebe duas listas como argumentos e takes two lists as arguments and é verdadeiro se:
  - O primeiro argumento é uma lista de a's, e
  - O segundo argumento é uma lista de b's, com exatamente o mesmo comprimento do primeiro

```
?- a2b([a,a,a,a],[b,b,b,b]).
true
?- a2b([a,a,a,a],[b,b,b]).
false
?- a2b([a,c,a,a],[b,b,b,t]).
false
```

Definindo a2b/2: passo 1

```
a2b([],[]).
```

- Muitas vezes o melhor modo de resolver tais problemas é pensar no caso mais simples possível.
- Aqui isto significa: a lista vazia

Definindo a2b/2: passo 2

```
a2b([],[]).
a2b([a|L1],[b|L2]):- a2b(L1,L2).
```

- Agora, pense recursivamente!
- Quando a2b/2 deveria decidir se duas listas não vazias são uma lista de a's e uma lista de b's com exatamente o mesmo tamanho?

Testando a2b/2

```
a2b([],[]).
a2b([a|L1],[b|L2]):- a2b(L1,L2).
```

```
?- a2b([a,a,a],[b,b,b]).
true
?-
```

Testando a2b/2

```
a2b([],[]).
a2b([a|L1],[b|L2]):- a2b(L1,L2).
```

```
?- a2b([a,a,a,a],[b,b,b]).
false
?-
```

Testando a2b/2

```
a2b([],[]).
a2b([a|L1],[b|L2]):- a2b(L1,L2).
```

```
?- a2b([a,t,a,a],[b,b,b,c]).
false
?-
```

Mais uma investigação sobre a2b/2

```
a2b([],[]).
a2b([a|L1],[b|L2]):- a2b(L1,L2).
```

```
?- a2b([a,a,a,a,a], X).
X = [b,b,b,b,b]

?-
```

Mais uma investigação sobre a2b/2

```
a2b([],[]).
a2b([a|L1],[b|L2]):- a2b(L1,L2).
```

```
?- a2b(X,[b,b,b,b,b,b,b]).
X = [a,a,a,a,a,a,a]

?-
```

Resumo desta aula

- Nesta aula introduzimos listas e predicados recursivos que operam sobre listas.
- O modo de programar ilustrado nestes predicados é fundamental no Prolog
- Muitos predicados Prolog que você escreverá serão variantes destes predicados.

Próxima aula

- Introdução à **aritmética** em Prolog
  - Introduzir a habilidade do Prolog na realização de operações aritméticas.
  - Aplicá-las a problemas simples de processameno de listas.
  - Introduzir a ideia de acumuladores.