

# Inteligência Artificial

## Aula 3 - vídeo 2 - Listas

26 de agosto de 2020

# Recursão em Prolog

Defina dois predicados  $par(Lista)$  e  $impar(Lista)$  de modo que eles são verdadeiros se seus argumento são, respectivamente, listas de tamanho par e ímpar.

# Recursão em Prolog

Defina dois predicados  $par(Lista)$  e  $impar(Lista)$  de modo que eles são verdadeiros se seus argumento são, respectivamente, listas de tamanho par e ímpar.

$par([ ])$ .

# Recursão em Prolog

Defina dois predicados *par(Lista)* e *impar(Lista)* de modo que eles são verdadeiros se seus argumento são, respectivamente, listas de tamanho par e ímpar.

*par*([ ]).

*impar*([\_]).

# Recursão em Prolog

Defina dois predicados *par(Lista)* e *impar(Lista)* de modo que eles são verdadeiros se seus argumento são, respectivamente, listas de tamanho par e ímpar.

```
par([ ]).
```

```
impar([_]).
```

```
par([X | Y]):- impar(Y).
```

# Recursão em Prolog

Defina dois predicados *par(Lista)* e *impar(Lista)* de modo que eles são verdadeiros se seus argumento são, respectivamente, listas de tamanho par e ímpar.

```
par([ ]).
```

```
impar([_]).
```

```
par([X | Y]):- impar(Y).
```

```
impar([X | Y]):- par(Y).
```

# Recursão em Prolog

Defina dois predicados *par(Lista)* e *impar(Lista)* de modo que eles são verdadeiros se seus argumento são, respectivamente, listas de tamanho par e ímpar.

```
par([ ]).
```

```
par([X | Y]):- impar(Y).
```

```
impar([_]).
```

```
impar([X | Y]):- par(Y).
```

# Recursão em Prolog

```
1 par([ ]).  
2 par([X | Y]):- impar(Y).  
3 impar([ ]).  
4 impar([X | Y]):- par(Y).
```

```
      ?- par([a,b,c,d]).  
      ↓  
X/a ,Y/[b,c,d]      2  
      ↓  
      ?- impar([b,c,d]).  
      ↓  
X/b ,Y/[c,d]      4  
      ↓  
      ?- par([c,d]).  
      ↓  
X/c ,Y/[d]      2  
      ↓  
      ?- impar([d]).  
      ↓  
X/d ,Y/[ ]      3  
      true
```

```
      ?- impar([a,b,c,d]).  
      ↓  
X/a ,Y/[b,c,d]      4  
      ↓  
      ?- par([b,c,d]).  
      ↓  
X/b ,Y/[c,d]      2  
      ↓  
      ?- impar([c,d]).  
      ↓  
X/c ,Y/[d]      4  
      ↓  
      ?- par([d]).  
      ↓  
X/d ,Y/[ ]      2  
      ↓  
      ?- impar([ ]).  
      false      3
```



# Recursão em Prolog

Dada uma lista  $L$ , faça um programa que determine se um certo elemento está presente ou não na lista.

# Recursão em Prolog

Dada uma lista L, faça um programa que determine se um certo elemento está presente ou não na lista.

**member(X,Y)**

# Recursão em Prolog

Dada uma lista  $L$ , faça um programa que determine se um certo elemento está presente ou não na lista.

**member(X,Y)** : *é verdade se o termo representado por X é um membro da lista representada por Y.*

# Recursão em Prolog

Dada uma lista L, faça um programa que determine se um certo elemento está presente ou não na lista.

**member(X,Y)** : *é verdade se o termo representado por X é um membro da lista representada por Y.*

? — *member*(3, [1, 2, 3, 4, 5]).  
true

? — *member*(6, [1, 2, 3, 4, 5]).  
false

? — *member*([3, 4], [1, 2, 3, 4, 5, [3, 4, 5], [3, 4]]).  
true

# Recursão em Prolog

**member(X,Y)** : é verdade se o termo representado por X é um membro da lista representada por Y.

## 1 Caso Base:

# Recursão em Prolog

**member(X,Y)** : é verdade se o termo representado por X é um membro da lista representada por Y.

- ❶ **Caso Base:** qual o caso mais fácil de identificar se um elemento está em uma lista?

# Recursão em Prolog

**member(X,Y)** : é verdade se o termo representado por X é um membro da lista representada por Y.

- ❶ **Caso Base:** qual o caso mais fácil de identificar se um elemento está em uma lista? *Se ele é o primeiro elemento da lista.*

# Recursão em Prolog

**member(X,Y)** : é verdade se o termo representado por X é um membro da lista representada por Y.

- ❶ **Caso Base:** qual o caso mais fácil de identificar se um elemento está em uma lista? *Se ele é o primeiro elemento da lista.*

X é um membro da lista que tem X como cabeça: *member(X, [X|\_]).*



# Recursão em Prolog

**member(X,Y)** : é verdade se o termo representado por X é um membro da lista representada por Y.

- 1 **Caso Base:** qual o caso mais fácil de identificar se um elemento está em uma lista? *Se ele é o primeiro elemento da lista.*

X é um membro da lista que tem X como cabeça: *member(X, [X|\_]).*

- 2 **Caso Geral:**

# Recursão em Prolog

**member(X,Y)** : é verdade se o termo representado por X é um membro da lista representada por Y.

- 1 **Caso Base:** qual o caso mais fácil de identificar se um elemento está em uma lista? *Se ele é o primeiro elemento da lista.*

X é um membro da lista que tem X como cabeça: *member(X, [X|\_]).*

- 2 **Caso Geral:** se o elementos que você quer não é o primeiro elemento da lista, onde ele pode estar?

# Recursão em Prolog

**member(X,Y)** : é verdade se o termo representado por X é um membro da lista representada por Y.

- 1 **Caso Base:** qual o caso mais fácil de identificar se um elemento está em uma lista? *Se ele é o primeiro elemento da lista.*

X é um membro da lista que tem X como cabeça: *member(X, [X|\_]).*

- 2 **Caso Geral:** se o elementos que você quer não é o primeiro elemento da lista, onde ele pode estar? *Na cauda!*

# Recursão em Prolog

**member(X,Y)** : é verdade se o termo representado por X é um membro da lista representada por Y.

- ❶ **Caso Base:** qual o caso mais fácil de identificar se um elemento está em uma lista? *Se ele é o primeiro elemento da lista.*

X é um membro da lista que tem X como cabeça: *member(X, [X|\_]).*

- ❷ **Caso Geral:** se o elementos que você quer não é o primeiro elemento da lista, onde ele pode estar? *Na cauda!*

X é um membro da lista se X é um membro da cauda da lista:  
*member(X, [\_|Y]) :- member(X, Y).*

# Recursão em Prolog - Exemplo

```
1 member(X,[X|_]).  
2 member(X,[_|L]) :- member(X,L).
```

```
?- member(c,[a,b,c,d,e]).
```

# Recursão em Prolog - Exemplo

```
1 member(X,[X|_]).  
2 member(X,[_|L]) :- member(X,L).
```

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



# Recursão em Prolog - Exemplo

```
1 member(X,[X|_]).  
2 member(X,[_|L]) :- member(X,L).
```

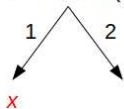
?- member(c,[a,b,c,d,e]).

1  
↙  
X

# Recursão em Prolog - Exemplo

```
1 member(X,[X|_]).  
2 member(X,[_|L]) :- member(X,L).
```

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

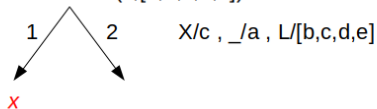




# Recursão em Prolog - Exemplo

```
1 member(X,[X|_]).  
2 member(X,[_|L]) :- member(X,L).
```

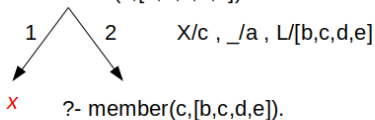
?- member(c,[a,b,c,d,e]).



# Recursão em Prolog - Exemplo

```
1 member(X,[X|_]).  
2 member(X,[_|L]) :- member(X,L).
```

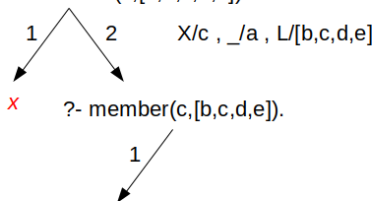
?- member(c,[a,b,c,d,e]).



# Recursão em Prolog - Exemplo

```
1 member(X,[X|_]).  
2 member(X,[_|L]) :- member(X,L).
```

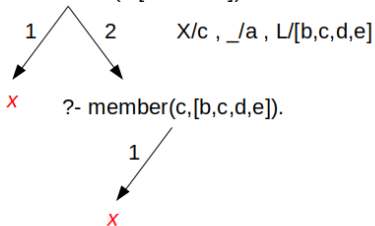
?- member(c,[a,b,c,d,e]).



# Recursão em Prolog - Exemplo

```
1 member(X,[X|_]).  
2 member(X,[_|L]) :- member(X,L).
```

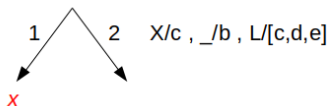
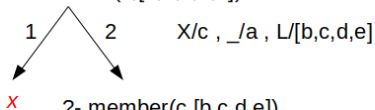
?- member(c,[a,b,c,d,e]).



# Recursão em Prolog - Exemplo

```
1 member(X,[X|_]).  
2 member(X,[_|L]) :- member(X,L).
```

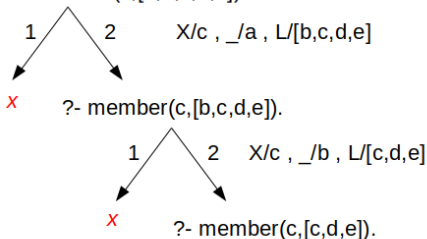
?- member(c,[a,b,c,d,e]).



# Recursão em Prolog - Exemplo

```
1 member(X,[X|_]).  
2 member(X,[_|L]) :- member(X,L).
```

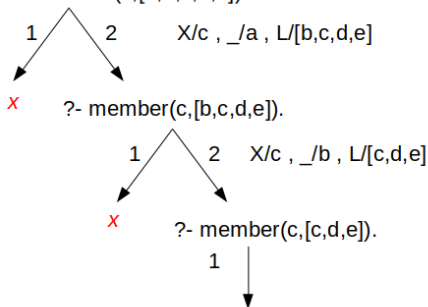
?- member(c,[a,b,c,d,e]).



# Recursão em Prolog - Exemplo

```
1 member(X,[X|_]).  
2 member(X,[_|L]) :- member(X,L).
```

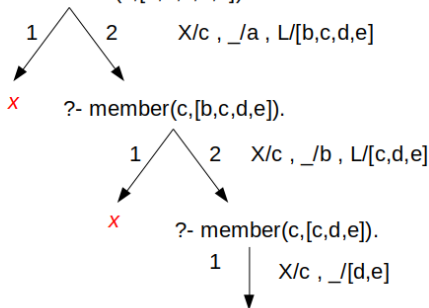
?- member(c,[a,b,c,d,e]).



# Recursão em Prolog - Exemplo

```
1 member(X,[X|_]).  
2 member(X,[_|L]) :- member(X,L).
```

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

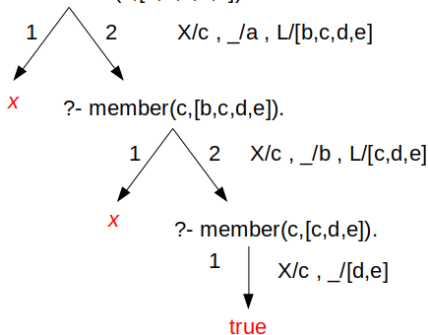




# Recursão em Prolog - Exemplo

```
1 member(X,[X|_]).  
2 member(X,[_|L]) :- member(X,L).
```

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



# Recursão em Prolog - Exemplo

```
1 member(X,[X|_]).  
2 member(X,[_|L]) :- member(X,L).
```

```
?- member(c,[1,2]).
```

# Recursão em Prolog - Exemplo

```
1 member(X,[X|_]).  
2 member(X,[_|L]) :- member(X,L).
```

?- member(c,[1,2]).



# Recursão em Prolog - Exemplo

```
1 member(X,[X|_]).  
2 member(X,[_|L]) :- member(X,L).
```

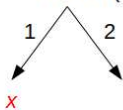
?- member(c,[1,2]).

1  
↙  
X

# Recursão em Prolog - Exemplo

```
1 member(X,[X|_]).  
2 member(X,[_|L]) :- member(X,L).
```

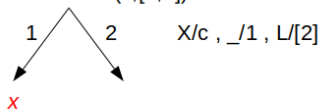
?- member(c,[1,2]).



# Recursão em Prolog - Exemplo

```
1 member(X,[X|_]).  
2 member(X,[_|L]) :- member(X,L).
```

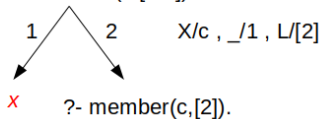
?- member(c,[1,2]).



# Recursão em Prolog - Exemplo

```
1 member(X,[X|_]).  
2 member(X,[_|L]) :- member(X,L).
```

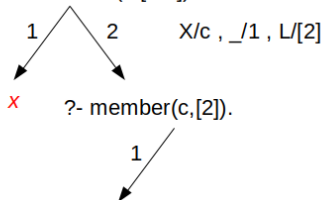
?- member(c,[1,2]).



# Recursão em Prolog - Exemplo

```
1 member(X,[X|_]).  
2 member(X,[_|L]) :- member(X,L).
```

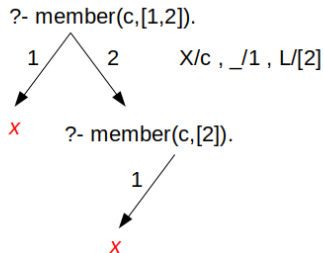
?- member(c,[1,2]).





# Recursão em Prolog - Exemplo

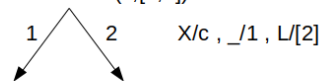
```
1 member(X,[X|_]).  
2 member(X,[_|L]) :- member(X,L).
```



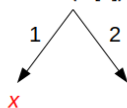
# Recursão em Prolog - Exemplo

```
1 member(X,[X|_]).  
2 member(X,[_|L]) :- member(X,L).
```

?- member(c,[1,2]).



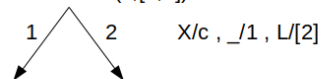
**X**      ?- member(c,[2]).



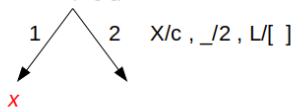
# Recursão em Prolog - Exemplo

```
1 member(X,[X|_]).  
2 member(X,[_|L]) :- member(X,L).
```

?- member(c,[1,2]).



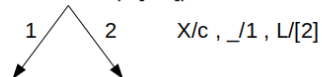
**X**    ?- member(c,[2]).



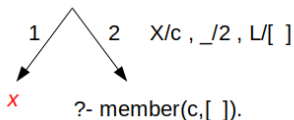
# Recursão em Prolog - Exemplo

```
1 member(X,[X|_]).  
2 member(X,[_|L]) :- member(X,L).
```

?- member(c,[1,2]).



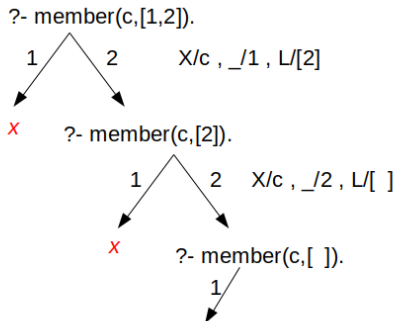
X ?- member(c,[2]).



X ?- member(c,[ ]).

# Recursão em Prolog - Exemplo

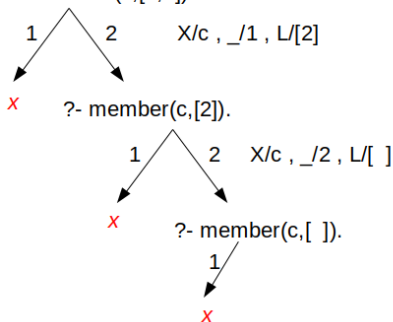
```
1 member(X,[X|_]).  
2 member(X,[_|L]) :- member(X,L).
```



# Recursão em Prolog - Exemplo

```
1 member(X,[X|_]).  
2 member(X,[_|L]) :- member(X,L).
```

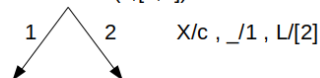
?- member(c,[1,2]).



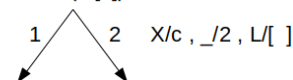
# Recursão em Prolog - Exemplo

```
1 member(X,[X|_]).  
2 member(X,[_|L]) :- member(X,L).
```

?- member(c,[1,2]).



?- member(c,[2]).

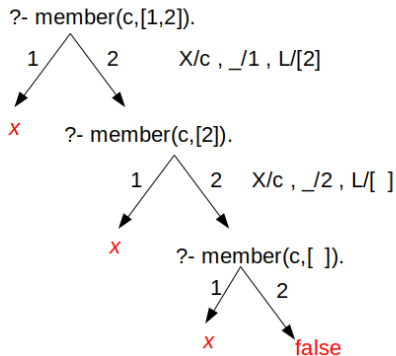


?- member(c,[ ]).



# Recursão em Prolog - Exemplo

```
1 member(X,[X|_]).  
2 member(X,[_|L]) :- member(X,L).
```





# Recursão em Prolog - Exemplo

```
1 member(X,[X|_]).  
2 member(X,[_|L]) :- member(X,L).
```

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

# Recursão em Prolog - Exemplo

```
1 member(X,[X|_]).  
2 member(X,[_|L]) :- member(X,L).
```

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

# Recursão em Prolog - Exemplo

```
1 member(X,[X|_]).  
2 member(X,[_|L]) :- member(X,L).
```

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




# Recursão em Prolog - Exemplo

```
1 member(X,[X|_]).  
2 member(X,[_|L]) :- member(X,L).
```

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

X/a , \_/[b,c]

1



# Recursão em Prolog - Exemplo

```
1 member(X,[X|_]).  
2 member(X,[_|L]) :- member(X,L).
```

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

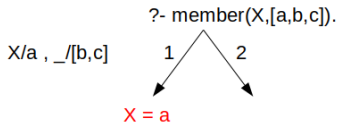
X/a , \_/[b,c]

1

X = a

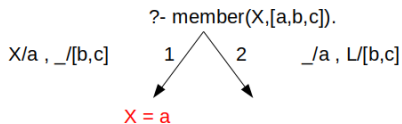
# Recursão em Prolog - Exemplo

```
1 member(X,[X|_]).  
2 member(X,[_|L]) :- member(X,L).
```



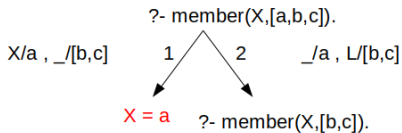
# Recursão em Prolog - Exemplo

```
1 member(X,[X|_]).  
2 member(X,[_|L]) :- member(X,L).
```



# Recursão em Prolog - Exemplo

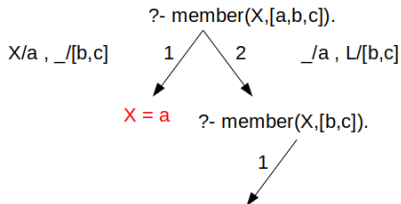
```
1 member(X,[X|_]).  
2 member(X,[_|L]) :- member(X,L).
```





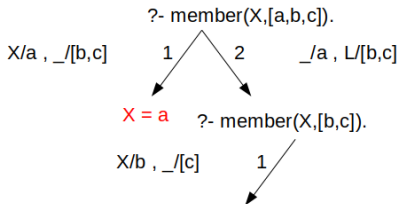
# Recursão em Prolog - Exemplo

```
1 member(X,[X|_]).  
2 member(X,[_|L]) :- member(X,L).
```



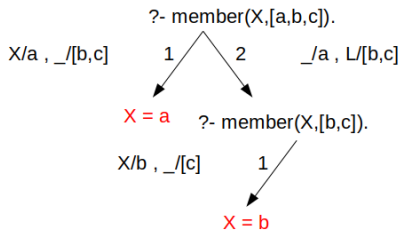
# Recursão em Prolog - Exemplo

```
1 member(X,[X|_]).  
2 member(X,[_|L]) :- member(X,L).
```



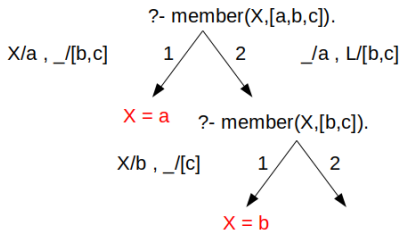
# Recursão em Prolog - Exemplo

```
1 member(X,[X|_]).  
2 member(X,[_|L]) :- member(X,L).
```



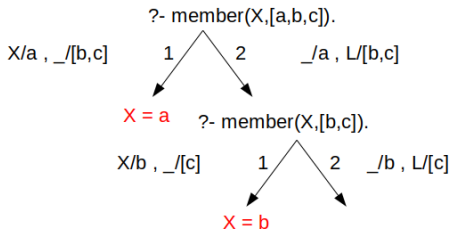
# Recursão em Prolog - Exemplo

```
1 member(X,[X|_]).  
2 member(X,[_|L]) :- member(X,L).
```



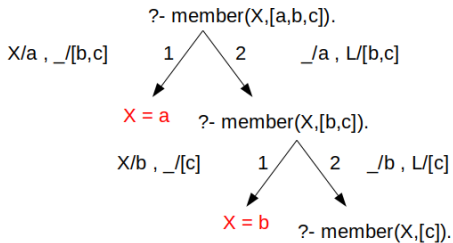
# Recursão em Prolog - Exemplo

```
1 member(X,[X|_]).  
2 member(X,[_|L]) :- member(X,L).
```



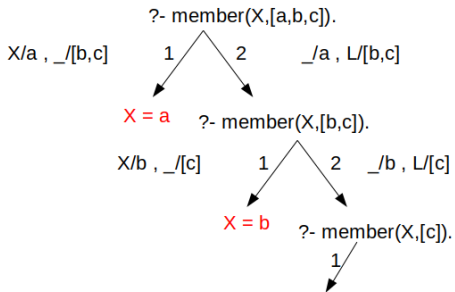
# Recursão em Prolog - Exemplo

```
1 member(X,[X|_]).  
2 member(X,[_|L]) :- member(X,L).
```



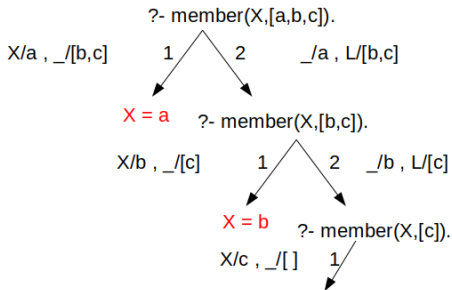
# Recursão em Prolog - Exemplo

```
1 member(X,[X|_]).  
2 member(X,[_|L]) :- member(X,L).
```



# Recursão em Prolog - Exemplo

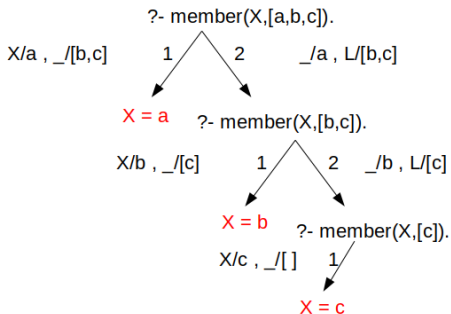
```
1 member(X,[X|_]).  
2 member(X,[_|L]) :- member(X,L).
```





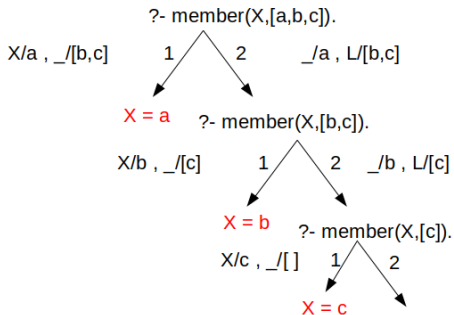
# Recursão em Prolog - Exemplo

```
1 member(X,[X|_]).  
2 member(X,[_|L]) :- member(X,L).
```



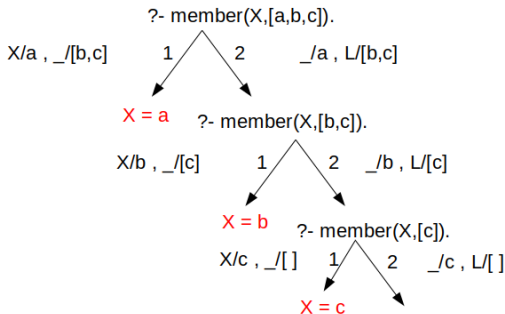
# Recursão em Prolog - Exemplo

```
1 member(X,[X|_]).  
2 member(X,[_|L]) :- member(X,L).
```



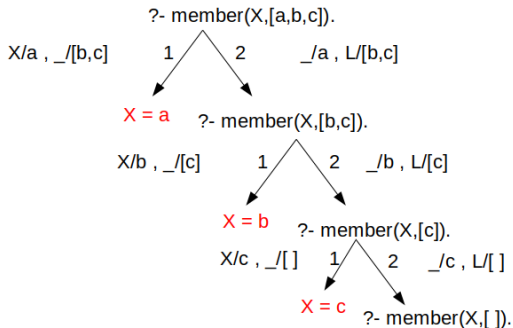
# Recursão em Prolog - Exemplo

```
1 member(X,[X|_]).  
2 member(X,[_|L]) :- member(X,L).
```



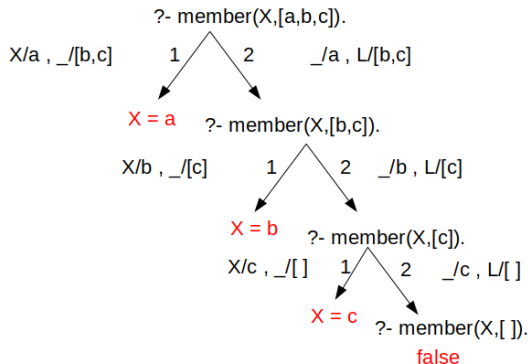
# Recursão em Prolog - Exemplo

```
1 member(X,[X|_]).  
2 member(X,[_|L]) :- member(X,L).
```



# Recursão em Prolog - Exemplo

```
1 member(X,[X|_]).  
2 member(X,[_|L]) :- member(X,L).
```



# Inteligência Artificial

## Aula 3 - vídeo 2 - Listas

26 de agosto de 2020