PROLOG

Rafael D. Ribeiro, M.Sc.
rafaeldiasribeiro@gmail.com
http://www.rafaeldiasribeiro.com.br

Listas

- Lista é uma sequência ordenada de uma quantidade qualquer de elementos.
- Os elementos contidos em uma lista devem ser separados por vírgulas, e precisam estar entre colchetes.
- Por exemplo, uma lista pode conter os nomes dos indivíduos, esta lista seria definida como:

[bia, joana, patricia, ana, antonio, bruno, joao]

Listas

- Existem dois tipos de listas, as listas vazias e as não vazias. Uma lista vazia é representada por [ ].
- Listas não vazias podem ser divididas em duas partes, são elas:
  - cabeça - corresponde ao primeiro elemento da lista;
  - cauda - corresponde aos elementos restantes da lista.

Listas

[bia, joana, patricia, ana, antonio, bruno, joao]

- cabeça - bia
- cauda - joana, patricia, ana, antonio, bruno, joao

- Observe que a cauda é uma nova lista, que por sua vez também possui cabeça e cauda. Assim, pode-se dizer que o último elemento de uma lista possui uma cauda vazia (uma lista vazia).

Listas

- Pode-se especificar que um elemento de uma lista é também uma lista, assim, pode-se representar listas tais como:

Hobbies1 = [tenis, musica].
Hobbies2 = [sky, comida].
Lista = [ana, Hobbies1, antonio,Hobbies2].

Listas

- É possível separar as partes de uma lista utilizando uma barra vertical, assim, pode-se escrever

Lista = [cabeça | cauda].

- Com isso, é possível determinar as seguintes listas:

[a | b, c] = [a, b, c]

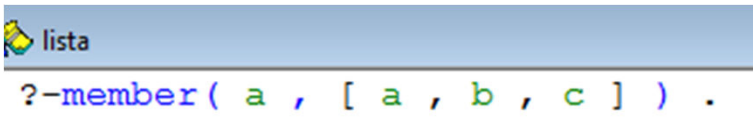
Operações sobre listas

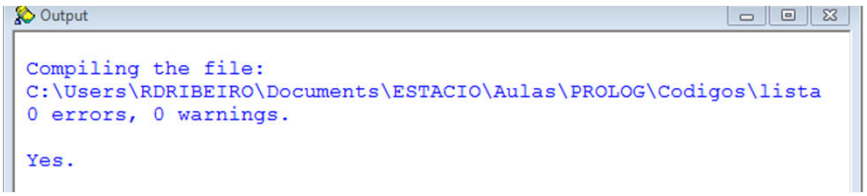
Checagem de pertinência

- Para se checar se um determinado elemento pertence à uma lista deve-se utilizar a relação member(x,y), que indica se x pertence à y, por exemplo:

```
3 ?- member(a, [a,b,c]).
Yes
4 ?- member(a, [[a,b],c]).
No
5 ?- member([a,b], [[a,b],c]).
Yes
```

Strawberry PROLOG





- "a" é um elemento da lista, uma vez que corresponde à cabeça desta.

Strawberry PROLOG

lista

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

    Compiling the fi
    C:\Users\RDRIBEI
    0 errors, 0 warn

    bYes.
    aYes.
    cYes.
    No.
```

Strawberry PROLOG

lista

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

Compiling the file:
C:\Users\RDRIBEIRO\Documents\ESTACIO\Aulas\PROLOG\Codigos\lista
0 errors, 0 warnings.

No.
```

- indica que “a” não pertence à lista, isso ocorre porque o elemento contido na lista é uma outra lista [a,b] e não o átomo “a”.

Strawberry PROLOG

lista

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

Output

```
Compiling the file:
C:\Users\RDRIBEIRO\Documents\ESTACIO\Aulas\PROLOG\Codigos\lista
0 errors, 0 warnings.

Yes.
```

Strawberry PROLOG

lista

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

```
Compiling t
C:\Users\RI
0 errors, (

[b,s]Yes.
[y,a]Yes.
cYes.
```

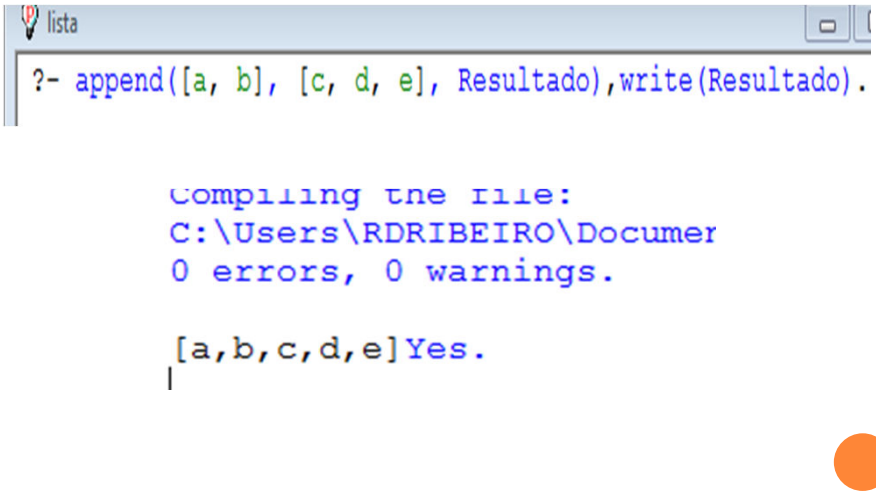
Operações sobre listas

- **Concatenação**
- Para realizar a concatenação de listas pode-se utilizar o predicado append(L1,L2,L3)., este predicado concatena a lista L1 e L2 exibindo o resultado em L3.
- O mesmo predicado pode ser utilizado para decompor listas

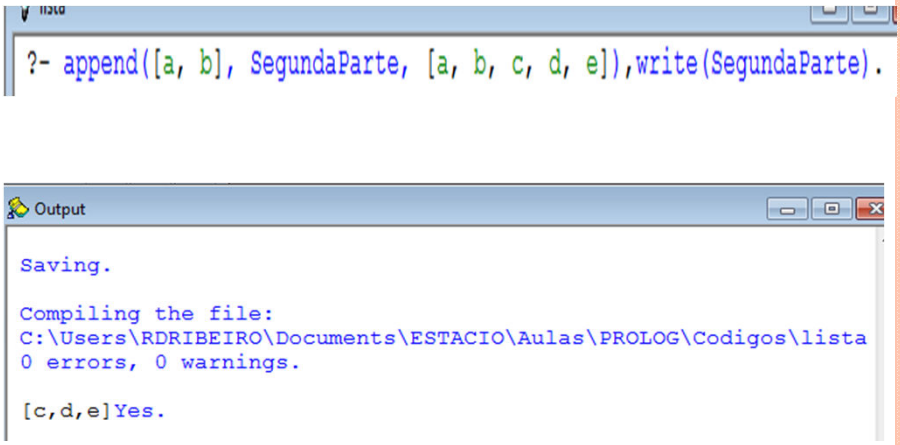
Operações sobre listas

- **Concatenação**
- Para definir a relação de concatenação, é necessário satisfazer as seguintes restrições:
  - um argumento é uma lista vazia - caso algum argumento seja vazio a concatenação resultará na repetição do argumento não vazio;
  - nenhum dos argumentos é vazio - a concatenação resulta na adição de todos os elementos da segunda lista ao final da primeira lista.

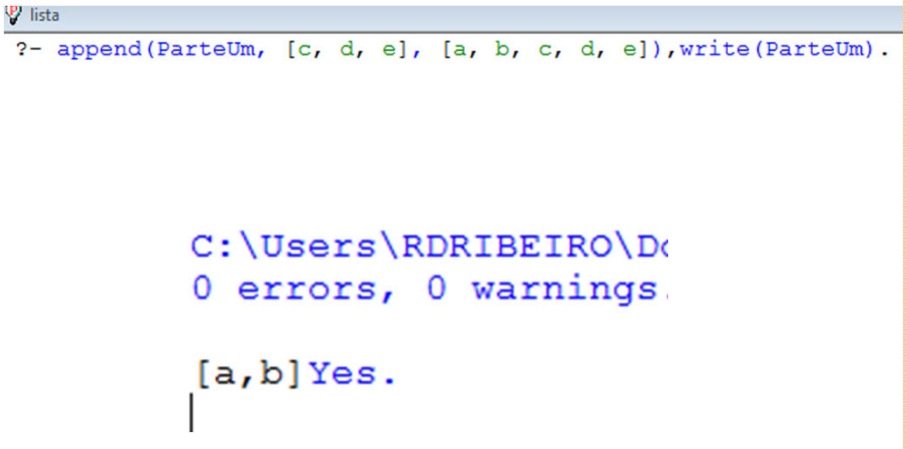
Strawberry PROLOG



Strawberry PROLOG



Strawberry PROLOG



Operações sobre listas

Inseri um Elemento

- A adição de um elemento à uma lista pode ser definida de modo simples. Para isso, basta inserir o elemento no início da lista, esta relação é definida através da seguinte regra:

```
insere(X, L, [X |L]).
```

Strawberry PROLOG

lista

```
exclui(X,[X|T],T).
?- exclui(a,[a,b,c],L),write(L).
```

Output

```
Compiling the file:
C:\Users\RDRIBEIRO\Documents
0 errors, 0 warnings.

[b,c]Yes.
```

Operações sobre listas

Excluindo um Elemento

- A exclusão de um elemento pode ser implementada através das seguintes regras:

```
exclui(X, [X | Tail], Tail).
exclui(X, [Y | Tail], [Y | Tail1]) :- exclui(X, Tail, Tail1).
```

Operações sobre listas

```
exclui(X, [X | Tail], Tail).
exclui(X, [Y | Tail], [Y | Tail1]) :- exclui(X, Tail, Tail1).
```

- a primeira regra é utilizada quando o elemento que se deseja excluir corresponde à cabeça da lista. Já a segunda regra exclui um elemento que pertence a cauda da lista.
- Vale salientar que esta implementação não exclui todos os elementos existentes na lista que correspondam ao elemento passado como argumento.

Strawberry PROLOG

```
insere(X,L,[X|L]).
?-insere(a,[b,c],X),write(X).
```

```
Compiling the file:
C:\Users\RDRIBEIRO\Documents
0 errors, 0 warnings.

[a,b,c]Yes.
```

Strawberry PROLOG

lista

```
exclui(X,[X|T],T).
exclui(X,[Y|T],[Y|T1]):-exclui(X,T,T1).

?- exclui(b,[a,b,c],L),write(L).
```

Output

```
Compiling the file:
C:\Users\RDRIBEIRO\Docu
0 errors, 0 warnings.

[a,c]Yes.
```

Strawberry PROLOG

lista

```
exclui(X,[X|T],T).
exclui(X,[Y|T],[Y|T1]):-exclui(X,T,T1).

?- exclui(c,[a,c,b,c],L),write(L).
```

Output

```
Saving.

Compiling the file:
C:\Users\RDRIBEIRO\Docu
0 errors, 0 warnings.

[a,b,c]Yes.
[a,c,b]Yes.
No.
```