Paradigmas de Linguagens de Programação - 2013.1 - Prof. Sergio D Zorzo
Aula 03 - Prolog - Exemplos

Exemplo 1
Programa pertence

1. Abrir o Prolog e criar uma novo programa

```
pertence(X, [X | Y]).
pertence(X, [Z | Y]) :- pertence(X,Y).
```

2. Fazer as seguintes consultas:

```
?- pertence(a, [1,2,a,c,b]).
?- pertence(a, [1,2,3]).
?- pertence(a, [1,2,3,[a, b, c], 4]).
?- pertence (X, [a,b,c]). % Recuperar todos os elementos de uma lista
?- trace.
?- pertence (X, [a,b,c]). % Para mostrar a busca
```

3. Modificar o programa para incluir variáveis anônimas
3.1. Explicar que o objetivo é indicar 2 problemas: (i) esquecimento de usar a variável em outro lugar; ou (ii) erro de digitação.

```
pertence(X, [X | _]).
pertence(X, [_ | Y]) :- pertence(X,Y).
```

Exemplo 2
Programa concatena

1. Abrir o Prolog e criar uma novo programa

```
conc([ ], L, L).
conc([X|L1], L2, [X|L3]) :- conc(L1, L2, L3).
```

2. Fazer as seguintes consultas:

```
?- conc([a, b, c], [1, 2, 3], L).
?- conc(L1, L2, [a, b, c]). % consulta invertida
?- conc(_, [Mes1, maio, Mes2| _],
[jan,fev,mar,abr,maio,jun,jul,ago,set,out,nov,dez]).
?- conc(L1, [E1,E2,E3], [a,b,c,[a,b],d,[ ]]).
```

Exemplo 3
Programa add_ultimo

1. Abrir o Prolog e criar uma novo programa

```
add_ultimo(X,[ ],[X]).
add_ultimo(X, [X1|Y],[X1|L]) :- add_ultimo(X,Y,L).
```

2. Fazer as seguintes consultas:

```
?- add_ultimo(x,[a,b,c],L).
?- add_ultimo([g,h],[f,d,i],L).
?- add_ultimo(X,Y,[a,b,c]).
```

Exemplo 4
Programa del

1. Abrir o Prolog e criar uma novo programa

```
del(X, [X|L], L).
del(X, [Y|L], [Y|L2]) :- del(X, L, L2).
```

2. Fazer as seguintes consultas:

```
?- del(a,[a,b,c],L).
?- del(a,[x,sf,fe,[d,a,c]],L).
```

Exemplo 5
Predicado "is"

1. Abrir o Prolog e executar as seguintes consultas:

```
?- X is 1+2.
?- Y is 1+5*(4-2).
?- X is 4/2.
?- X is 1+2.
?- 2 is 4/2.
?- 1 is sin(pi/2).
?- 1.0 is sin(pi/2).
?- Y is 1+2, X is Y.
?- Y is X, X is 1+2.
```

Exemplo 6
Predicados aritméticos

1. Abrir o Prolog e executar as seguintes consultas:

```
?- 2+1 < 6-2.
?- 2+1 > (8/4)+5.
?- 1+2 is 2+1.
?- 1+2 =:= 2+1.
?- 1+2 =:= X.
?- 1 =:= sin(pi/2).
```

Exemplo 7
Predicado "="

1. Abrir o Prolog e executar as seguintes consultas:

```
?- 5 = 5.
?- fred = fred.
?- X = Y
?- pai_de(joao,paulo) = pai_de(X,Y).
?- X = 2+5.
?- 2+5 = X.
?- 2+5 = X, Y is X + 3.
?- X is 2+5.
```

```
?- 2+5 is X.
?- 1+2 = 2+1.
?- 1+X = 1+2.
?- 1+2 =:= 2+1.
```

Negar as consultas acima

```
?- 5 \= 5.
?- fred \= fred.
?- X \= Y
?- pai_de(joao,paulo) \= pai_de(X,Y).
?- X \= 2+5.
?- 2+5 \= X.
```

Exemplo 8
Predicado "=="

1. Abrir o Prolog e executar as seguintes consultas:

```
?- nome == nome.
?- X == X.
?- X == 5.
?- X == Y.
?- pred(1) == pred(X).
?- pred(1,teste(X)) == pred(1,teste(X)).
?- pred(1,teste(X)) == pred(teste(X),1).
?- +(1,2) == +(2,1).
?- 1+2 == 2+1.
?- 1+2 =:= 2+1.
?- +(1,2) =:= 2+1.
```

Exemplo 9
Ordenação de termos

1. Abrir o Prolog e executar as seguintes consultas:

```
?- X @< a.
?- a @< 2.
?- X @< Y.
?- Y is 2, X is 4, X @< Y.
?- 2+3 < 6.
?- 2+3 @< 6.
?- boneca @< carro.
?- boneca @< barco.
?- boneca(zzzz) @< carro(aaaa,bbb).
?- boneca(zzzz) @< boneca(aaaa,bbb).
?- boneca(zzzz,zzzz) @< boneca(aaaa,bbb)
?- boneca(a,a,a) @< boneca(aaaa,bbb)
?- 3+4 @< 4+3.
?- +(3,4) @< +(4,3).
```

Exemplo 10
Uso do "corte" em Prolog

1. Abrir o Prolog e criar um novo programa

```
f(X,0) :- X < 3.
f(X,2) :- 3 =< X, X < 6.
f(X,4) :- 6 =< X.
```

2. Fazer as seguintes consultas:

```
?- trace.
?- f(1,Y).
```

3. Adicionar os predicados de corte e refazer a consulta

```
f(X,0) :- X < 3, !.
f(X,2) :- 3 =< X, X < 6, !.
f(X,4) :- 6 =< X.
```

4. Mostrar que para a consulta f(7,Y) não há diferença
5. Modificar o programa para aumentar ainda mais a eficiência

```
f(X,0) :- X < 3, !.
f(X,2) :- X < 6, !.
f(_,4).
```

6. Criar o seguinte programa em Prolog

```
mmc(X, Y, Z) :- positivo(Z),
            Z mod X =:= 0,
            Z mod Y =:= 0,
            !.
positivo(1).
positivo(Z) :- positivo(W), Z is W+1.
```

7. Rodar as seguintes consultas

```
?- notrace.
?- mmc(12,15,Z).
?- mmc(12,15,60).
?- mmc(12,15,120).
```

Exemplo 11
Demonstrar o uso de verificação de tipo

1. Fazer o seguinte programa Prolog

```
desparentize([], []).
desparentize([X|Y], [X|Z]) :- \+ is_list(X),
      desparentize(Y, Z), !.
desparentize([X|Y], Z) :- desparentize(X, X1),
      desparentize(Y, Y1),
      conc(X1, Y1, Z).
conc([ ], L, L).
conc([X|L1], L2, [X|L3]) :- conc(L1, L2, L3).
```

2. Rodar as seguintes consultas

```
desparentize([a,[b],c,[d,[e]],f],L).
desparentize([a,[b],c,[], [[d,[e]],f]],L).
desparentize([a],L).
desparentize([],L).
desparentize(a,L).
desparentize(funtor(a,b),L).
```

Exemplo 12
Modificando a base de dados

1. Abrir o Prolog e criar um novo programa

```
estudante('Joao').
estudante('Marcia').
estudante('Paulo').
faltas('Joao',5).
faltas('Marcia',2).
```

2. Fazer as seguintes consultas:

```
?- estudante(X).
?- faltas('Joao',X).
?- abolish(faltas/2).
?- abolish([estudante/1, faltas/2]).
?- abolish(faltas,2).
```

3. Fazer as seguintes consultas:

```
?- assert(gosta(maria,cinema)).
?- assert(gosta(joao,computacao)).
?- assert((gosta(maria,Pessoa) :- gosta(Pessoa,computacao))).
```

Exemplo 13
Entrada e saída

1. Abrir o Prolog e fazer a seguinte consulta

```
?- read(X), Y is X+1, write('Resultado'), nl, write(Y).
```

Exemplo 14
Predicados sem argumentos e com o mesmo nome

1. Abrir o Prolog e criar um novo programa

```
soma :- write('Digite uma lista de numeros'),
        read(Lista),
        soma(Lista,Resultado),
        soma(Resultado, Resultado, Dobro),
        write('O dobro da soma dos elementos da lista e = '),
        write(Dobro),
        nl.
```

```
soma([],0).
soma([Elem|Cauda], S) :- soma(Cauda,S1),
                         S is S1 + Elem.

soma(X, Y, Z) :- Z is X + Y.
```

2. Fazer a seguinte consulta

```
?- soma.
```

Exemplo 15
Meta-variáveis

---

1. Abrir o Prolog e criar um novo programa

```
conta([],0).
conta([_|Cauda],N) :- conta(Cauda,N1), N is N1 + 1.
soma([],0).
soma([Elem|Cauda],S) :- soma(Cauda,S1), S is S1 + Elem.

programa1 :- write('Entre com a lista de elementos'),
             read(Lista),
             conta(Lista,N),
             write('O numero de elementos e '),
             nl, write(N), nl.

programa2 :- write('Entre com a lista de numeros'),
             read(Lista),
             soma(Lista,N),
             write('A soma dos elementos e '),
             nl, write(N), nl.

principal :- write('Digite o nome do programa a ser executado - programa1
ou programa2 - '),
             read(NP),
             NP,
             write(NP), nl.

programa(X) :- X, write(X), nl.
```

2. Fazer as seguintes consultas

```
?- principal.
?- programa(conta([1,2,3],N)).
```