Paradigmas de Programação

Prof. Maicon R. Zatelli

Prolog - Programação Lógica
Introdução

Universidade Federal de Santa Catarina
Florianópolis - Brasil
2018/1

Prolog

Prolog (PROgramming in LOGic) - 1972

Lógica e declarativa

Criada por Alain Colmerauer e Philippe Roussel.

Linguagem puramente lógica e baseada num subconjunto do cálculo de predicados de primeira ordem, o que é definido por cláusulas de Horn. Alguns conceitos fundamentais são unificação, recursão, e backtracking.

Programação Declarativa

Em linguagens procedurais, o programador especifica o passo a passo, em detalhes, de como resolver um problema. Já, na programação declarativa, o programador apenas diz "qual é o problema" e deixa que o sistema da linguagem faça o resto.

Assim, em Prolog, declara-se uma meta para ser atingida e o sistema do Prolog trabalha para atingí-la.

Aplicações da linguagem Prolog

- Banco de dados inteligentes
- Processamento de linguagem natural
- Sistemas especialistas
- Especificação de linguagens
- Aprendizado de máquina
- Planejamento (*planning*)
- Raciocínio automatizado
- Resolução de problemas

Prolog - Compilando e Executando

Faça o download do SWI-Prolog e instale em seu computador:

- `http://www.swi-prolog.org/`

Para compilar use o seguinte comando:

- swipl -o hello -c hello.pl

Para executar, basta executar: ./hello

Prolog - swipl

Swipl é o cliente Prolog.

Pode ser aberto por meio do comando `swipl`

Por meio do swipl, pode-se diretamente interpretar/executar código Prolog.

Para carregar um arquivo prolog no swipl, usa-se a seguinte expressão

- ['/home/paradigmas/prolog/arquivo.pl'].

Prolog

Considerados brancos: espaços, <cr>, <lf>, tabulações, comentários.

Comentário de linha: %

Comentário de bloco: /* */

Prolog - Relações

Em prolog especifica-se relacionamentos entre objetos e também propriedades de objetos.

Quando diz-se que Bob possui uma bicicleta, cria-se uma relação de posse entre dois objetos: Bob e bicicleta.

Quando pergunta-se "Bob possui a bicicleta?", tenta-se descobrir o relacionamento.

Pode-se especificar relacionamentos da seguinte forma: duas pessoas são irmãos se ambas são homens e possuem genitores em comum.

Prolog - Fatos, Regras e Consultas

Fatos descrevem relações explícitas entre objetos e também propriedades explícitas que objetos possuem.

- Bob é homem.
- Bob é genitor de Tom.

Fatos

```
mulher(pam).
mulher(liz).
homem(tom).
homem(bob).
genitor(pam, bob).
genitor(tom, bob).
genitor(tom, liz).
```

- Identificadores de relacionamentos são denominados **predicados** e identificadores de objetos são denominados **átomos**. Ambos devem iniciar com letra minúscula.
  - Predicados: homem, genitor, mulher
  - Átomos: bob, tom, liz, pam
- **Aridade** é o número de parâmetros que há em cada *functor*.

Prolog - Fatos, Regras e Consultas

Regras permitem descobrir relações mesmo sem as relações estarem explicitamente declaradas como fatos, ou seja, regras definem relações implícitas entre objetos e também propriedades implícitas que objetos possuem.

- Ex: X é pai de Y se X é genitor de Y e X é homem.

Regras: `cabeça :- corpo`

```
pai(X,Y) :- genitor(X,Y), homem(X).
mae(X,Y) :- genitor(X,Y), mulher(X).
```

- and = , ou conjunção
- or = ; ou disjunção
- not = **not**
- if = :-
- Palavras iniciadas em maiúsculo representam variáveis.
  - _ é variável anônima, que unifica com qualquer coisa.

Prolog - Fatos, Regras e Consultas

Consultas são "perguntas" feitas sobre relacionamentos entre objetos e propriedades de objetos.

- Ex: quem é o pai de Liz?

Consultas

```
pai(X,liz).
```

Prolog - Fatos, Regras e Consultas

Consultas são "perguntas" feitas sobre relacionamentos entre objetos e propriedades de objetos.

- Ex: quem é o pai de Liz?

Consultas

```
pai(X,liz).
X = tom.
```

Prolog - Fatos, Regras e Consultas

Consultas são "perguntas" feitas sobre relacionamentos entre objetos e propriedades de objetos.

- Ex: quem é o pai de Liz?

Consultas

```
pai(X,liz).
X = tom.

?- pai(X,Y).
```

Prolog - Fatos, Regras e Consultas

Consultas são "perguntas" feitas sobre relacionamentos entre objetos e propriedades de objetos.

- Ex: quem é o pai de Liz?

Consultas

```
pai(X,liz).
X = tom.

?- pai(X,Y).
X = tom,
Y = bob
```

Prolog - Fatos, Regras e Consultas

Consultas são "perguntas" feitas sobre relacionamentos entre objetos e propriedades de objetos.

- Ex: quem é o pai de Liz?

Consultas

```
pai(X,liz).
X = tom.

?- pai(X,Y).
X = tom,
Y = bob ;
X = tom,
Y = liz
```

Prolog - Fatos, Regras e Consultas

Consultas são "perguntas" feitas sobre relacionamentos entre objetos e propriedades de objetos.

- Ex: quem é o pai de Liz?

Consultas

```
pai(X,liz).
X = tom.

?- pai(X,Y).
X = tom,
Y = bob ;
X = tom,
Y = liz ;
X = bob,
Y = ana
```

Prolog - Fatos, Regras e Consultas

Consultas são "perguntas" feitas sobre relacionamentos entre objetos e propriedades de objetos.

- Ex: quem é o pai de Liz?

Consultas

```
pai(X,liz).
X = tom.

?- pai(X,Y).
X = tom,
Y = bob ;
X = tom,
Y = liz ;
X = bob,
Y = ana ;
X = bob,
Y = pat
```

Prolog - Fatos, Regras e Consultas

Consultas são "perguntas" feitas sobre relacionamentos entre objetos e propriedades de objetos.

- Ex: quem é o pai de Liz?

Consultas

```
pai(X,liz).
X = tom.

?- pai(X,Y).
X = tom,
Y = bob ;
X = tom,
Y = liz ;
X = bob,
Y = ana ;
X = bob,
Y = pat ;
false.
```

Prolog - Fatos, Regras e Consultas

Regras: cabeça :- corpo

```
avo(X, Y) :- genitor(Z, Y), genitor(X, Z), homem(X).
irmao(X,Y) :- genitor(Z, X), genitor(Z, Y), X \== Y, homem(X).
irma(X,Y) :- genitor(Z, X), genitor(Z, Y), X \== Y, mulher(X).
irmaos(X,Y) :- (irmao(X,Y); irma(X,Y)), X \== Y.
```

- X é avô de Y se existe um Z tal que Z é genitor de Y e X é genitor de Z e X é homem.
- X é irmão de Y se existe um Z tal que Z é genitor de X e Z é genitor de Y e X é diferente de Y e X é homem.
- X é irmã de Y se existe um Z tal que Z é genitor de X e Z é genitor de Y e X é diferente de Y e X é mulher.
- X e Y são irmãos se X é irmão de Y ou X é irmã de Y e X é diferente de Y.

Prolog - Fatos, Regras e Consultas

- Quem é o avô de Pat?
- Quais são todos os avôs na árvore genealógica?

Consultas

```
?- avo(X,pat).
```

Prolog - Fatos, Regras e Consultas

- Quem é o avô de Pat?
- Quais são todos os avôs na árvore genealógica?

Consultas

```
?- avo(X,pat).
X = tom.
```

Prolog - Fatos, Regras e Consultas

- Quem é o avô de Pat?
- Quais são todos os avôs na árvore genealógica?

Consultas

```
?- avo(X,pat).
X = tom.

?- avo(X,Y).
```

Prolog - Fatos, Regras e Consultas

- Quem é o avô de Pat?
- Quais são todos os avôs na árvore genealógica?

Consultas

```
?- avo(X,pat).
X = tom.

?- avo(X,Y).
X = tom,
Y = ana
```

Prolog - Fatos, Regras e Consultas

- Quem é o avô de Pat?
- Quais são todos os avôs na árvore genealógica?

Consultas

```
?- avo(X,pat).
X = tom.

?- avo(X,Y).
X = tom,
Y = ana ;
X = tom,
Y = pat
```

Prolog - Fatos, Regras e Consultas

- Quem é o avô de Pat?
- Quais são todos os avôs na árvore genealógica?

Consultas

```
?- avo(X,pat).
X = tom.

?- avo(X,Y).
X = tom,
Y = ana ;
X = tom,
Y = pat ;
X = tom,
Y = bill
```

Prolog - Fatos, Regras e Consultas

- Quem é o avô de Pat?
- Quais são todos os avôs na árvore genealógica?

Consultas

```
?- avo(X,pat).
X = tom.

?- avo(X,Y).
X = tom,
Y = ana ;
X = tom,
Y = pat ;
X = tom,
Y = bill ;
X = bob,
Y = jim.
```

- Digite `trace.` antes de iniciar uma consulta para verificar o passo a passo da execução.
- Digite `notrace.` para terminar a execução passo a passo.

Prolog - Fatos, Regras e Consultas

- Quem é o avô de Pat?
- Quais são todos os avôs na árvore genealógica?

Consultas

```
?- avo(X,pat).
X = tom.

?- avo(X,Y).
X = tom,
Y = ana ;
X = tom,
Y = pat ;
X = tom,
Y = bill ;
X = bob,
Y = jim.
```

- Digite `trace.` antes de iniciar uma consulta para verificar o passo a passo da execução.
- Digite `notrace.` para terminar a execução passo a passo.

Prolog - Estruturas

Uma estrutura pode ser representada por meio de vários argumentos.

Estruturas - Fatos

```
ponto(1.3, 22.1).
triangulo(ponto(1, 2), ponto(5, 2), ponto(7, 7)).
data(20, maio, 2018).
fatoHitorico(data(22, abril, 1500), 'Descobrimento do Brasil').
segmentoReta(ponto(1, 2), ponto(5, 2)).
```

Estruturas - Regras

```
vertical(segmentoReta(ponto(X, Y1), ponto(X, Y2))) :- Y1 \== Y2.

horizontal(segmentoReta(ponto(X1, Y), ponto(X2, Y))) :- X1 \== X2.

obliqua(segmentoReta(ponto(X1, Y1), ponto(X2, Y2))) :- X1 \== X2,
                                                       Y1 \== Y2.
```

- Faça algumas consultas...

Prolog - Estruturas

Consultas

```
?- vertical(segmentoReta(ponto(1,2), ponto(7,2))).
false.

?- horizontal(segmentoReta(ponto(1,2), ponto(7,2))).
true.

?- obliqua(segmentoReta(ponto(1,2), ponto(7,3))).
true.
```

Prolog - Estruturas

Descobrir todos os segmentos de reta formados pelos pontos.

```
ponto(1.3, 22.1).
ponto(2, 1).
ponto(3, 4).

segmentoReta(ponto(1, 2), ponto(5, 2)).
segmentoReta(ponto(X1,Y1), ponto(X2, Y2)) :-
  ponto(X1,Y1),
  ponto(X2, Y2),
  ponto(X1,Y1) \== ponto(X2, Y2).
```

Prolog - Estruturas

Consulta

```
?- segmentoReta(X,Y).
```

Prolog - Estruturas

Consulta

```
?- segmentoReta(X,Y).
X = ponto(1, 2),
Y = ponto(5, 2)
```

Prolog - Estruturas

Consulta

```
?- segmentoReta(X,Y).
X = ponto(1, 2),
Y = ponto(5, 2) ;
X = ponto(1.3, 22.1),
Y = ponto(2, 1)
```

Prolog - Estruturas

Consulta

```
?- segmentoReta(X,Y).
X = ponto(1, 2),
Y = ponto(5, 2) ;
X = ponto(1.3, 22.1),
Y = ponto(2, 1) ;
X = ponto(1.3, 22.1),
Y = ponto(3, 4)
```

Prolog - Estruturas

Consulta

```
?- segmentoReta(X,Y).
X = ponto(1, 2),
Y = ponto(5, 2) ;
X = ponto(1.3, 22.1),
Y = ponto(2, 1) ;
X = ponto(1.3, 22.1),
Y = ponto(3, 4) ;
X = ponto(2, 1),
Y = ponto(1.3, 22.1)
```

Prolog - Estruturas

Consulta

```
?- segmentoReta(X,Y).
X = ponto(1, 2),
Y = ponto(5, 2) ;
X = ponto(1.3, 22.1),
Y = ponto(2, 1) ;
X = ponto(1.3, 22.1),
Y = ponto(3, 4) ;
X = ponto(2, 1),
Y = ponto(1.3, 22.1) ;
X = ponto(2, 1),
Y = ponto(3, 4)
```

Prolog - Estruturas

Consulta

```
?- segmentoReta(X,Y).
X = ponto(1, 2),
Y = ponto(5, 2) ;
X = ponto(1.3, 22.1),
Y = ponto(2, 1) ;
X = ponto(1.3, 22.1),
Y = ponto(3, 4) ;
X = ponto(2, 1),
Y = ponto(1.3, 22.1) ;
X = ponto(2, 1),
Y = ponto(3, 4) ;
X = ponto(3, 4),
Y = ponto(1.3, 22.1)
```

Prolog - Estruturas

Consulta

```
?- segmentoReta(X,Y).
X = ponto(1, 2),
Y = ponto(5, 2) ;
X = ponto(1.3, 22.1),
Y = ponto(2, 1) ;
X = ponto(1.3, 22.1),
Y = ponto(3, 4) ;
X = ponto(2, 1),
Y = ponto(1.3, 22.1) ;
X = ponto(2, 1),
Y = ponto(3, 4) ;
X = ponto(3, 4),
Y = ponto(1.3, 22.1) ;
X = ponto(3, 4),
Y = ponto(2, 1)
```

Prolog - Estruturas

Consulta

```
?- segmentoReta(X,Y).
X = ponto(1, 2),
Y = ponto(5, 2) ;
X = ponto(1.3, 22.1),
Y = ponto(2, 1) ;
X = ponto(1.3, 22.1),
Y = ponto(3, 4) ;
X = ponto(2, 1),
Y = ponto(1.3, 22.1) ;
X = ponto(2, 1),
Y = ponto(3, 4) ;
X = ponto(3, 4),
Y = ponto(1.3, 22.1) ;
X = ponto(3, 4),
Y = ponto(2, 1) ;
false.
```

Prolog

Alguns predicados importantes

- true/0 = retorna sempre verdade.
- false/0 = retorna sempre falso.
- fail/0 = retorna sempre falso (equivalente a `false`).
- halt/0 = termina a execução.

Outros serão vistos mais adiante...

Prolog - Operadores e Funções

Aritméticos

- +, -, *, /, mod, ^, sqrt

Exemplos

```
X is mod(7, 2)
Y is 8 / 3
K is 8 ^ 2
A is sqrt(121)
```

Prolog - Operadores e Funções

Lógicos

- , (and/conjunção)
- ; (or/disjunção)
- not (not)

Exemplos

```
genitor(X,Y) :- pai(X,Y); mae(X,Y).
nand(X,Y) :- not(X);not(Y).
```

- genitor(X,Y): X é genitor de Y se X é pai de Y ou X é mãe de Y.
- nand(X,Y): operação da porta lógica `NAND`.

Prolog - Operadores e Funções

Relacionais

- =:= (igualdade aritmética), =\= (desigualdade aritmética)
- >, >=, =<, >
- == (identidade de termos), \== (não identidade de termos)
- = (unificação), \= (não unificação)
- is (unificação)

Exemplos

```
?- 4 =:= 2 + 2.
```

Prolog - Operadores e Funções

Relacionais

- =:= (igualdade aritmética), =\= (desigualdade aritmética)
- >, >=, =<, <
- == (identidade de termos), \== (não identidade de termos)
- = (unificação), \= (não unificação)
- is (unificação)

Exemplos

```
?- 4 =:= 2 + 2.
true.
?- 8 ^ 2 =\= 64.
```

Prolog - Operadores e Funções

Relacionais

- =:= (igualdade aritmética), =\= (desigualdade aritmética)
- >, >=, =<, >
- == (identidade de termos), \== (não identidade de termos)
- = (unificação), \= (não unificação)
- is (unificação)

Exemplos

```
?- 4 =:= 2 + 2.
true.
?- 8 ^ 2 =\= 64.
false.
?- 4 =< 14 / 2.
```

Prolog - Operadores e Funções

Relacionais

- =:= (igualdade aritmética), =\= (desigualdade aritmética)
- >, >=, =<, >
- == (identidade de termos), \== (não identidade de termos)
- = (unificação), \= (não unificação)
- is (unificação)

Exemplos

```
?- 4 =:= 2 + 2.
true.
?- 8 ^ 2 =\= 64.
false.
?- 4 =< 14 / 2.
true.
?- 4 >= 14 / 2.
```

Prolog - Operadores e Funções

Relacionais

- =:= (igualdade aritmética), =\= (desigualdade aritmética)
- >, >=, =<, >
- == (identidade de termos), \== (não identidade de termos)
- = (unificação), \= (não unificação)
- is (unificação)

Exemplos

```
?- 4 =:= 2 + 2.
true.
?- 8 ^ 2 =\= 64.
false.
?- 4 =< 14 / 2.
true.
?- 4 >= 14 / 2.
false.
```

Prolog - Operadores e Funções

```
?- 'a' == a.
```

Prolog - Operadores e Funções

```
?- 'a' == a.
true.
?- X == a.
```

Prolog - Operadores e Funções

```
?- 'a' == a.
true.
?- X == a.
false.
?- X = a.
```

Prolog - Operadores e Funções

```
?- 'a' == a.
true.
?- X == a.
false.
?- X = a.
X = a.
?- 3 = 3.
```

Prolog - Operadores e Funções

```
?- 'a' == a.
true.
?- X == a.
false.
?- X = a.
X = a.
?- 3 = 3.
true.
?- 'a' = a.
```

Prolog - Operadores e Funções

```
?- 'a' == a.
true.
?- X == a.
false.
?- X = a.
X = a.
?- 3 = 3.
true.
?- 'a' = a.
true.
?- nome(maior) == nome(X).
```

Prolog - Operadores e Funções

```
?- 'a' == a.
true.
?- X == a.
false.
?- X = a.
X = a.
?- 3 = 3.
true.
?- 'a' = a.
true.
?- nome(maior) == nome(X).
false.
?- nome(maior) = nome(X).
```

Prolog - Operadores e Funções

```
?- 'a' == a.
true.
?- X == a.
false.
?- X = a.
X = a.
?- 3 = 3.
true.
?- 'a' = a.
true.
?- nome(maior) == nome(X).
false.
?- nome(maior) = nome(X).
X = maior.
?- nome(mario) = nome(mario).
```

Prolog - Operadores e Funções

```
?- 'a' == a.
true.
?- X == a.
false.
?- X = a.
X = a.
?- 3 = 3.
true.
?- 'a' = a.
true.
?- nome(maior) == nome(X).
false.
?- nome(maior) = nome(X).
X = maior.
?- nome(mario) = nome(mario).
true.
?- nome(mario) == nome(mario).
```

Prolog - Operadores e Funções

```
?- 'a' == a.
true.
?- X == a.
false.
?- X = a.
X = a.
?- 3 = 3.
true.
?- 'a' = a.
true.
?- nome(maior) == nome(X).
false.
?- nome(maior) = nome(X).
X = maior.
?- nome(mario) = nome(mario).
true.
?- nome(mario) == nome(mario).
true.
?- 5 = 3 + 2.
```

Prolog - Operadores e Funções

```
?- 'a' == a.
true.
?- X == a.
false.
?- X = a.
X = a.
?- 3 = 3.
true.
?- 'a' = a.
true.
?- nome(maior) == nome(X).
false.
?- nome(maior) = nome(X).
X = maior.
?- nome(mario) = nome(mario).
true.
?- nome(mario) == nome(mario).
true.
?- 5 = 3 + 2.
false.
?- 5 is 3 + 2.
```

Prolog - Operadores e Funções

```
?- 'a' == a.
true.
?- X == a.
false.
?- X = a.
X = a.
?- 3 = 3.
true.
?- 'a' = a.
true.
?- nome(maior) == nome(X).
false.
?- nome(maior) = nome(X).
X = maior.
?- nome(mario) = nome(mario).
true.
?- nome(mario) == nome(mario).
true.
?- 5 = 3 + 2.
false.
?- 5 is 3 + 2.
true.
```

Prolog - Operadores e Funções

```
?- 5 =:= 3 + 2.
```

Prolog - Operadores e Funções

```
?- 5 =:= 3 + 2.
true.
?- 5 == 3 + 2.
```

Prolog - Operadores e Funções

```
?- 5 =:= 3 + 2.
true.
?- 5 == 3 + 2.
false.
?- 5 = 3 + 2.
```

Prolog - Operadores e Funções

```
?- 5 =:= 3 + 2.
true.
?- 5 == 3 + 2.
false.
?- 5 = 3 + 2.
false.
?- 5 is 3 + 2.
```

Prolog - Operadores e Funções

```
?- 5 =:= 3 + 2.
true.
?- 5 == 3 + 2.
false.
?- 5 = 3 + 2.
false.
?- 5 is 3 + 2.
true.
?- X == 3 + 2.
```

Prolog - Operadores e Funções

```
?- 5 =:= 3 + 2.
true.
?- 5 == 3 + 2.
false.
?- 5 = 3 + 2.
false.
?- 5 is 3 + 2.
true.
?- X == 3 + 2.
false.
?- X = 3 + 2.
```

Prolog - Operadores e Funções

```
?- 5 =:= 3 + 2.
true.
?- 5 == 3 + 2.
false.
?- 5 = 3 + 2.
false.
?- 5 is 3 + 2.
true.
?- X == 3 + 2.
false.
?- X = 3 + 2.
X = 3+2.
?- X is 3 + 2.
```

Prolog - Operadores e Funções

```
?- 5 =:= 3 + 2.
true.
?- 5 == 3 + 2.
false.
?- 5 = 3 + 2.
false.
?- 5 is 3 + 2.
true.
?- X == 3 + 2.
false.
?- X = 3 + 2.
X = 3+2.
?- X is 3 + 2.
X = 5.
?- 3+2 = 3 + 2.
```

Prolog - Operadores e Funções

```
?- 5 =:= 3 + 2.
true.
?- 5 == 3 + 2.
false.
?- 5 = 3 + 2.
false.
?- 5 is 3 + 2.
true.
?- X == 3 + 2.
false.
?- X = 3 + 2.
X = 3+2.
?- X is 3 + 2.
X = 5.
?- 3+2 = 3 + 2.
true.
?- 3+2 == 3 + 2.
```

Prolog - Operadores e Funções

```
?- 5 =:= 3 + 2.
true.
?- 5 == 3 + 2.
false.
?- 5 = 3 + 2.
false.
?- 5 is 3 + 2.
true.
?- X == 3 + 2.
false.
?- X = 3 + 2.
X = 3+2.
?- X is 3 + 2.
X = 5.
?- 3+2 = 3 + 2.
true.
?- 3+2 == 3 + 2.
true.
?- 3+2 is 3 + 2.
```

Prolog - Operadores e Funções

```
?- 5 =:= 3 + 2.
true.
?- 5 == 3 + 2.
false.
?- 5 = 3 + 2.
false.
?- 5 is 3 + 2.
true.
?- X == 3 + 2.
false.
?- X = 3 + 2.
X = 3+2.
?- X is 3 + 2.
X = 5.
?- 3+2 = 3 + 2.
true.
?- 3+2 == 3 + 2.
true.
?- 3+2 is 3 + 2.
false.
?- 3 + 2 =:= 3 + 2.
```

Prolog - Operadores e Funções

```
?- 5 =:= 3 + 2.
true.
?- 5 == 3 + 2.
false.
?- 5 = 3 + 2.
false.
?- 5 is 3 + 2.
true.
?- X == 3 + 2.
false.
?- X = 3 + 2.
X = 3+2.
?- X is 3 + 2.
X = 5.
?- 3+2 = 3 + 2.
true.
?- 3+2 == 3 + 2.
true.
?- 3+2 is 3 + 2.
false.
?- 3 + 2 =:= 3 + 2.
true.
```

Prolog - Operadores e Funções

- = e == não avaliam expressões.
- **is** avalia a expressão à direita. Normalmente é usado com uma variável do lado esquerdo.
- =:= avalia ambas as expressões (à direita e à esquerda).

Prolog - Funções

Podemos especificar uma função `max` como uma regra por meio de um predicado de aridade 3 `max(X,Y,Max)`, onde `Max` deve armazenar o maior entre `X` e `Y`.

```
max(X,Y,X) :- X >= Y.
max(X,Y,Y) :- Y > X.
```

- X é o máximo se X é maior ou igual a Y.
- Y é o máximo se Y é maior que X.

```
?- max(3,4,X).
X = 4.

?- max(5,4,X).
X = 5 .
```

Prolog - Funções/Predicados Básicos

Concatena átomos e strings

```
atom_concat('paradigmas ', 'ufsc', X).
string_concat("paradigmas ", "ufsc", X).
```

Prolog - Recursividade

```
genitor(pam, bob).
genitor(tom, bob).
genitor(tom, liz).
genitor(bob, ana).
genitor(bob, pat).
genitor(liz,bill).
genitor(pat, jim).

ascendente(X,Y) :- genitor(X,Y).
ascendente(X,Y) :- genitor(X, Z), ascendente(Z, Y).
```

Consultas

```
?- ascendente(X, pat).
```

Prolog - Recursividade

```
genitor(pam, bob).
genitor(tom, bob).
genitor(tom, liz).
genitor(bob, ana).
genitor(bob, pat).
genitor(liz,bill).
genitor(pat, jim).

ascendente(X,Y) :- genitor(X,Y).
ascendente(X,Y) :- genitor(X, Z), ascendente(Z, Y).
```

Consultas

```
?- ascendente(X, pat).
X = bob
```

Prolog - Recursividade

```
genitor(pam, bob).
genitor(tom, bob).
genitor(tom, liz).
genitor(bob, ana).
genitor(bob, pat).
genitor(liz,bill).
genitor(pat, jim).

ascendente(X,Y) :- genitor(X,Y).
ascendente(X,Y) :- genitor(X, Z), ascendente(Z, Y).
```

Consultas

```
?- ascendente(X, pat).
X = bob ;
X = pam
```

Prolog - Recursividade

```
genitor(pam, bob).
genitor(tom, bob).
genitor(tom, liz).
genitor(bob, ana).
genitor(bob, pat).
genitor(liz,bill).
genitor(pat, jim).

ascendente(X,Y) :- genitor(X,Y).
ascendente(X,Y) :- genitor(X, Z), ascendente(Z, Y).
```

Consultas

```
?- ascendente(X, pat).
X = bob ;
X = pam ;
X = tom
```

Prolog - Recursividade

```
genitor(pam, bob).
genitor(tom, bob).
genitor(tom, liz).
genitor(bob, ana).
genitor(bob, pat).
genitor(liz,bill).
genitor(pat, jim).

ascendente(X,Y) :- genitor(X,Y).
ascendente(X,Y) :- genitor(X, Z), ascendente(Z, Y).
```

Consultas

```
?- ascendente(X, pat).
X = bob ;
X = pam ;
X = tom ;
false.
```

Prolog

Definindo uma meta inicial.

```
:- initialization(main). % define a meta inicial
main :- writeln('Hello World!'),
        halt.
```

Tente executar este código em:

http://www.compileonline.com/execute_prolog_online.php

Prolog - Entrada e Saída

Entrada

read(X) lê uma palavra, número ou string.

- Caso houverem espaços, dê a entrada como 'palavra1 palavra2', ou seja, uma string formada por mais de uma palavra deve estar entre apóstrofos.

Saída

writeln(X) imprime e quebra linha

write(X) imprime e não quebra linha

tab(K) imprime K tabulações

nl pula linha

Prolog - Entrada e Saída

Exemplo

```
:- initialization(main).

main :- read(X),
        writeln(X),
        Y is sqrt(X),
        writeln(Y),
        halt.
```

Prolog - Fatorial

```
:- initialization(main).
fatorial(1,1).
fatorial(N,K) :- N1 is N - 1, fatorial(N1, K1), K is N * K1.

main :- write('Informe um numero '),
        read(K),
        write('Fatorial de '),
        write(K),
        write(' eh '),
        fatorial(K, X),
        writeln(X),
        halt.
```

```
?- ['/home/maicon/fatorial.pl'].
% /home/maicon/fatorial.pl compiled 0.00 sec, 6 clauses
Informe um numero 5.
Fatorial de 5 eh 120
```

Tente executar este código em:

http://www.compileonline.com/execute_prolog_online.php

Prolog - Alguns Links Úteis

- `http://lpn.swi-prolog.org/lpnpage.php?pageid=online`
- `http://www.swi-prolog.org/pldoc/doc/_CWD_/index.html`

Prolog

Ver atividade no Moodle