Taller Prolog

¿Qué haces para aprender un lenguaje de programación?

Jacinto Dávila

Taller Prolog para programadores tradicionales

¿Cómo aprender un lenguaje de programación?

1. Hola Mundo
2. Estructuras de datos
3. Estructura de un archivo fuente
4. Estructuras de programación
   1. Procedimiento, función o método
   2. Secuencia de instrucciones
   3. Estructuras de decisión
   4. Estructuras de repetición
      1. Incondicional
      2. Condicional
5. Totalmente nuevo
6. Ver código existente

Hola Mundo en Prolog

```
?- write('Hola Mundo').
Hola Mundo
True.

?- swritef(S, '%15L%w', ['Hello', 'World']).
S = "'Hello'        World".
```

Estructuras de Datos

NO se declaran!

Simplemente se usan

Ejemplos:
Constantes: `1, a, 'A'`
Variables: (Cualquier identificador que comience con mayúscula) `X, Y, A, Jacinto, N_de_m`
Funciones: `f(X), [1,2,3]`

Estructura de un archivo fuente

Puede ser sólo CLAUSULAS
(Ver siguiente lámina)

Aunque se permite invocaciones

Por ejemplo:

```
| :- use_module(library(lists), [ member/2,                    |
|                                 append/2 as list_concat      |
|                               ]).                            |
```

Procedimiento, función o método

```
metodo(Arg1, Arg2, Arg3) :-
  sub_pro1(Arg1, Arg4),
  subpro2(Arg2, Arg4, Arg3).
```

```
natural(0).
natural(X) :- Y is X – 1, natural(Y).
```

Secuencia de instrucciones

Después del :- cualquier secuencia de predicados separados por coma y que termina (la secuencia) con un punto.

```
procedimiento :-
        paso1,
        paso2,
        paso3,
        paso4.
```

Estructuras de decisión

```
metodo :-
   paso_previo,
   ( condicion -> opcion1 ; opcion 2 ),
   paso_posterior.
                              --
metodo :-
  paso_previo, condicion, opcion1, paso_posterior.
metodo :-
  paso_previo, not(condicion), opcion2, paso_posterior.

                              --

metodo :- paso_previo, decision, paso_posterior.

decision :- condicion, opcion1.
decision :- opcion2.
```

Estructuras de Repetición Incondicional

Esto es parece a un *for* convencional, pero cuidado!

```
?- forall(member(Result = Formula, [2 = 1 + 1, 4 = 2 * 2]), Result =:= Formula).
True.
```

También estos son parecidos:

```
?- findall(X, member(X, [2,4,5]), L).
L = [2, 4, 5].

?- foreach(between(1,3,X), dif(X,Y)).
dif(Y, 3),
dif(Y, 2),
dif(Y, 1),
dif(Y, 1),
dif(Y, 2),
dif(Y, 1),
dif(Y, 1).
```

Estructuras de Repetición Incondicional

Pero si quiero un contador:

```
longitud([], 0).
longitud([_|R], N) :-
  longitud(R, NR), N is NR + 1.
```

```
?- length([a,b,c,d], L).
L = 4.
```

Estructuras de Repetición Condicional

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

ancestro(X,Y) :- padre(X,Y).
ancestro(X,Y) :- padre(X,Z),
  ancestro(Z,Y).
```

Totalmente Nuevo

UNIFICACION:

```
?- X = cualquier_cosa.
X = cualquier_cosa.

?- [X|Y] = [1,2,3,4].
X = 1,
Y = [2, 3, 4].
```

Especialmente bueno para

Metaprogramación

http://resumidor.sourceforge.net
http://gloria.sourceforge.net
http://galatea.sourceforge.net