

**INSTITUTO POLITÉCNICO NACIONAL**  
**ESCUELA SUPERIOR DE CÓMPUTO**



**Artificial Intelligence**

## **Tarea No. 2**

# Prolog

Profesor: **Hernández Cruz Macario**

**Manzano León Jansel**

**3CM4**

Fecha de entrega:  
**15-Octubre-2012**

# Introducción

## II. SINTAXIS GENERAL.

- **Programa Prolog:** Conjunto de predicados/declaraciones (hechos y reglas) representando los conocimientos que poseemos en un determinado dominio o campo de nuestra competencia.

Comentarios	/* ... */	
Predicados	nombre(term1, ..., termN).	Aridad=nº Argumentos
<b>Términos</b>		
Constantes Simbólicas	Ejemplos: a, x, '2', juan, "camisa" (1ª letra en minúsculas).	
Constantes Numéricas	Ejemplos: 2, 355, -1	
Variables	Ejemplos: X, Y, Nombre (1ª letra literal en mayúsculas).	
Variable Anónima "_"	Su valor es indiferente.	
Estructura (Función)	Functor(arg1, ..., argN).	

## III. PROCESADOR DE PROLOG

- **Unificación:** Proceso de localizar patrones que "emparejen" términos.
- **Instanciación:** Asignación temporal de valores a variables para permitir la unificación.
- **Retroceso:** Cuando fracasa la unificación de un predicado vuelta atrás y ensayo de otra unificación.
- **Procesador de Objetivos:** para cada subobjetivo (de izquierda a derecha), llama al procesador de reglas.
- **Procesador de Reglas:** Explora las cláusulas (hechos y reglas) de arriba a bajo buscando unificaciones.

## IV. REGLAS Y HECHOS

Hecho	Predicado( ... ).
-------	-------------------

Regla	Consecuente :- Antecedente
-------	----------------------------

- Formato de Cláusula de Horn.
- Afirmación general sobre los objetos y sus relaciones.
- La definición de la regla puede incluir hechos y otras reglas.
- Un predicado puede venir definido por varias reglas.
- El orden de las reglas determinan el orden en que se encuentran las soluciones.
- Las conjunciones se establecen mediante la coma ",".

## VIII. LISTAS

Representación	
[elem1, ... elemN]	Secuencia de elementos separados por coma y entre corchetes.
[]	Lista vacía
[cab cola]	Estructura con dos componentes: cabeza lista y el resto de la lista.

- Construcción  $Z = [a | Y]$ . si  $Y = [b, c] \rightarrow Z = [a, b, c]$
- Destrucción  $[X | Y] = Z$ . si  $Z = [a, b, c] \rightarrow X = a, Y = [b, c]$

# Desarrollo

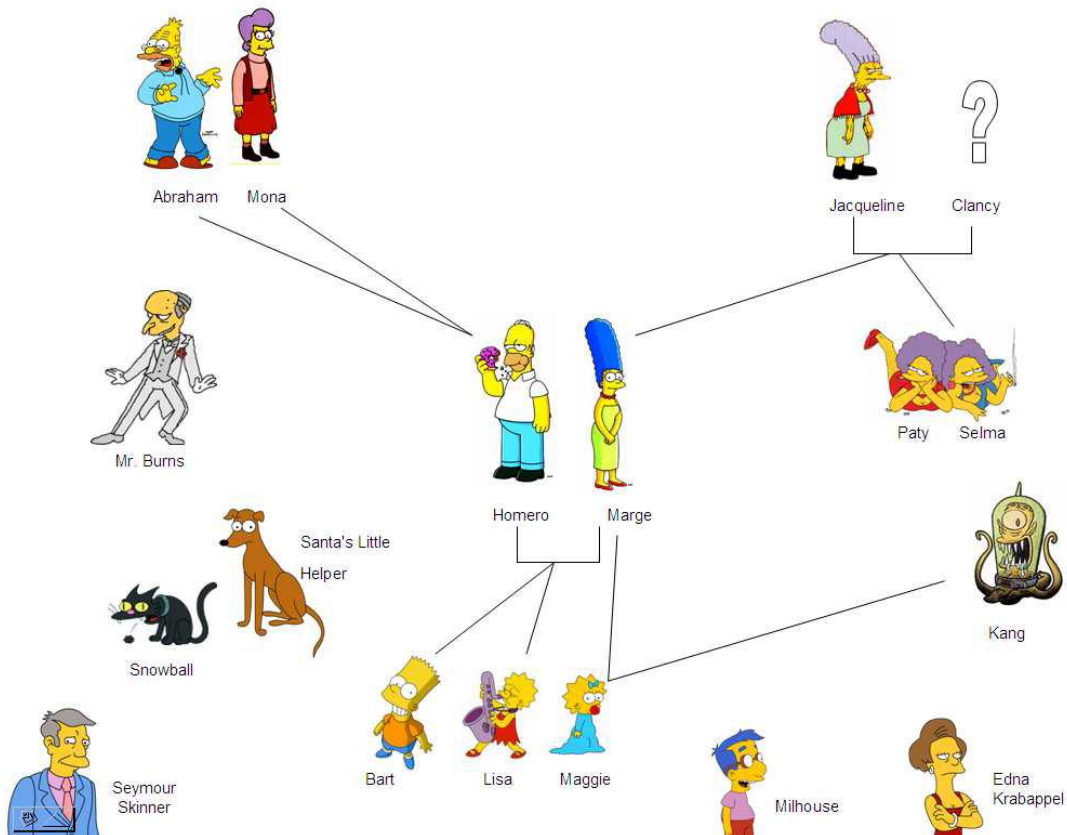
## I. Introducción a Prolog – Hechos y Reglas

### Ejercicio 1.1

Considerando la lámina anexa (tomada de la serie de televisión “Los Simpsons”), represente en lenguaje Prolog las características de los objetos y las relaciones entre ellos.

Asimismo, basadas en las relaciones “progenitor” y el género de las personas (si es hombre o mujer), establecer las reglas para:

```
abuelo(X,Y):-  
abuela(X,Y):-  
tio(X,Y):-  
tia(X,Y):-  
hermano(X,Y):-  
hermana(X,Y):-
```



## Solución (1.1)

```
macho(homero).  
macho(bart).  
macho(abraham).  
macho(mr_burns).  
macho(clancy).  
macho(seymour_skinner).  
macho(milhouse).
```

```
hembra(jacqueline).  
hembra(edna).  
hembra(mona).  
hembra(marge).  
hembra(paty).  
hembra(selma).  
hembra(maggie).  
hembra(lisa).
```

```
animal(snowball).
```

```
alien(kang).  
macho(kang).
```

```
progenitor(homero,bart).  
progenitor(marge,bart).
```

```
progenitor(homero,lisa).  
progenitor(marge,lisa).
```

```
progenitor(kang,maggie).  
progenitor(marge,maggie).
```

```
progenitor(abraham,homero).  
progenitor(mona,homero).
```

```
progenitor(clancy,marge).  
progenitor(jacqueline,marge).
```

```
progenitor(clancy,selma).  
progenitor(jacqueline,selma).
```

```
progenitor(clancy,paty).  
progenitor(jacqueline,paty).
```

```
conyuge(homero,marge).
```

```
abuelo(X,Y):-progenitor(M,Y),progenitor(X,M),macho(X).  
abuela(X,Y):-progenitor(M,Y),progenitor(X,M),hembra(X).
```

```
hermano(X,Y):-progenitor(M,Y),progenitor(M,X),macho(X).  
hermana(X,Y):-progenitor(M,Y),progenitor(M,X),hembra(X).
```

```
tio(X,Y):-progenitor(M,Y),hermano(M,X).  
tia(X,Y):-progenitor(M,Y),hermana(M,X).
```

```
SWI-Prolog -- c:/Users/NuTcrAcKeR/Documents/ESCOM/5º/IA/tarea2/tarea2.pro
File Edit Settings Run Debug Help
23 ?- abuelo(X,lisa).
X = abraham ;
X = clancy ;
false.

24 ?- abuelo(clancy,lisa).
true.

25 ?- abuela(X,bart).
X = mona ;
X = jacqueline.

26 ?- abuelo(X,maggie).
X = clancy ;
false.

27 ?- hermano(bart,lisa).
true.

28 ?- hermano(selma,paty).
false.

29 ?- hermana(selma,paty).
true.

30 ?- hermana(X,bart).
X = lisa ;
X = lisa ;
X = maggie.

31 ?- hermana(lisa,maggie).
true.

39 ?- tia(selma,bart).
true.

40 ?- tia(paty,bart).
true.

41 ?- tia(homero,salma).
false.

42 ?- tio(salma,lisa).
false.

43 ?- tia(paty,lisa).
true.
```

## II. Aritmética y recursividad en Prolog

### Ejercicio 2.1

Considere la definición de la sucesión de Fibonacci.

$$fib(n) = n \text{ si } n \leq 1$$

$$fib(n) = fib(n-1) + fib(n-2) \text{ si } n > 1$$

De tal forma que los primeros términos de la sucesión son:

0, 1, 1, 2, 3, 5, 8, 13, 21...

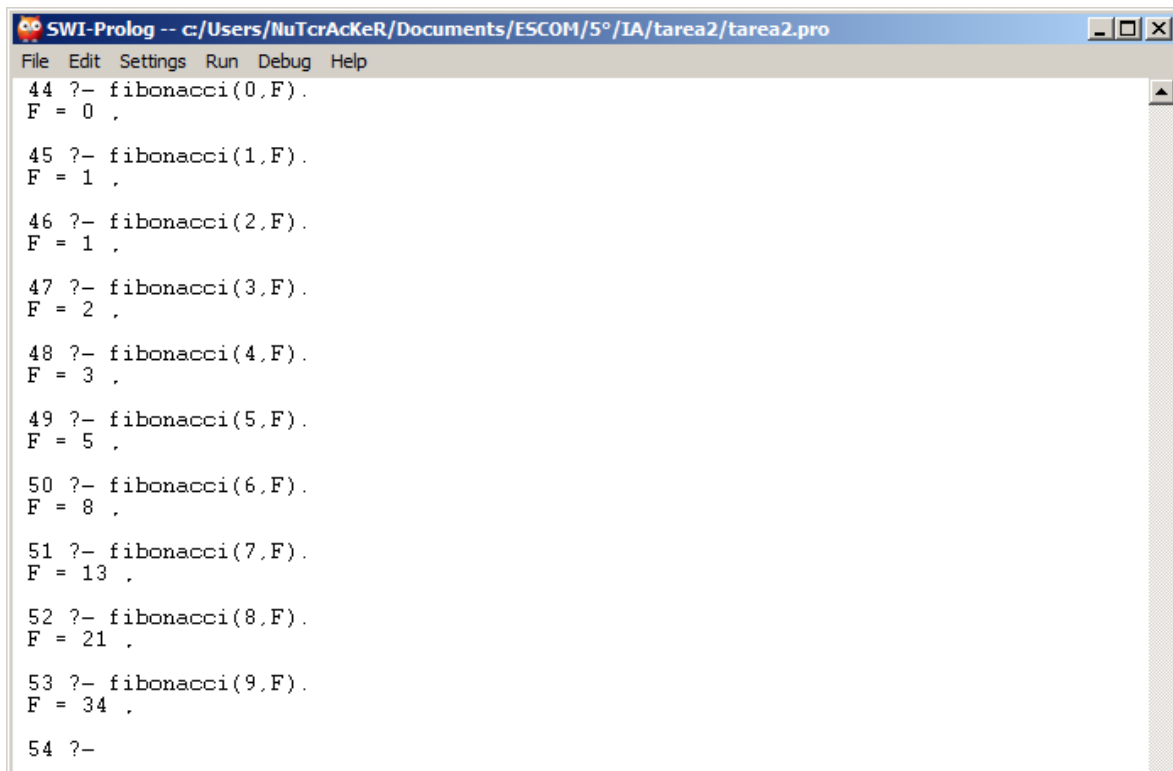
Desarrolle un conjunto de reglas para calcular el término n-esimo de la sucesión.

```
?- fibonacci(0,F).
F = 0
?- fibonacci(1,F).
F = 1
?- fibonacci(2,F).
```

```
F = 1
?- fibonacci(3,F).
F = 2
?- fibonacci(4,F).
F = 3
```

### Solución (2.1)

```
fibonacci(0,0).
fibonacci(1,1).
fibonacci(N,F):- N1 is N-1,N2 is N1-1,
                 fibonacci(N1,A),
                 fibonacci(N2,B),
                 F is A+B.
```



The screenshot shows the SWI-Prolog IDE window titled "SWI-Prolog -- c:/Users/NuTcrAcKeR/Documents/ESCOM/5º/IA/tarea2/tarea2.pro". The menu bar includes File, Edit, Settings, Run, Debug, and Help. The main text area contains the following Prolog queries and their results:

```
44 ?- fibonacci(0,F).
F = 0 ,

45 ?- fibonacci(1,F).
F = 1 ,

46 ?- fibonacci(2,F).
F = 1 ,

47 ?- fibonacci(3,F).
F = 2 ,

48 ?- fibonacci(4,F).
F = 3 ,

49 ?- fibonacci(5,F).
F = 5 ,

50 ?- fibonacci(6,F).
F = 8 ,

51 ?- fibonacci(7,F).
F = 13 ,

52 ?- fibonacci(8,F).
F = 21 ,

53 ?- fibonacci(9,F).
F = 34 ,

54 ?-
```

## III. Listas

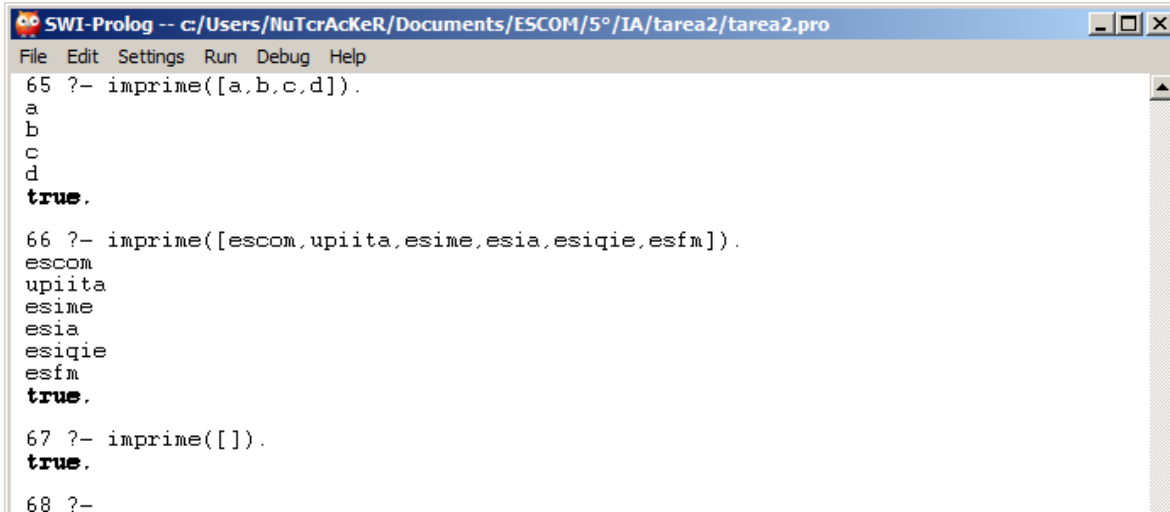
### Ejercicio 3.1

Escriba la(s) regla(s) necesaria(s) para imprimir una lista de la siguiente forma.

```
?- imprime([a, b, c, d]).
a
b
c
d
true
```

### Solución (3.1)

```
imprime([]).  
imprime([H|T]):- write_ln(H),  
                  imprime(T).
```



```
SWI-Prolog -- c:/Users/NuTcrAcKeR/Documents/ESCOM/5°/IA/tarea2/tarea2.pro  
File Edit Settings Run Debug Help  
65 ?- imprime([a,b,c,d]).  
a  
b  
c  
d  
true.  
66 ?- imprime([escom,upiita,esime,esia,esiqie,esfm]).  
escom  
upiita  
esime  
esia  
esiqie  
esfm  
true.  
67 ?- imprime([]).  
true.  
68 ?-
```

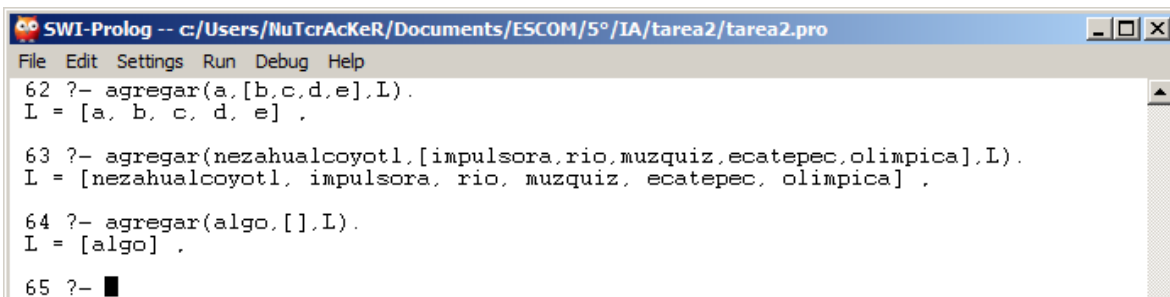
### Ejercicio 3.2

Escriba la(s) regla(s) necesaria(s) para agregar un elemento al principio de una lista.

```
?- agregar(gato,[zorro, zopilote, puerco],L).  
L = [gato, zorro, zopilote, puerco]  
true
```

### Solución (3.2)

```
agregar(E,L,[E|L]).  
agregar(E,[X|Y],[X|Z]):-agregar(E,Y,Z).
```



```
SWI-Prolog -- c:/Users/NuTcrAcKeR/Documents/ESCOM/5°/IA/tarea2/tarea2.pro  
File Edit Settings Run Debug Help  
62 ?- agregar(a,[b,c,d,e],L).  
L = [a, b, c, d, e] .  
63 ?- agregar(nezahualcoyotl,[impulsora,rio,muzquiz,ecatepec,olimpica],L).  
L = [nezahualcoyotl, impulsora, rio, muzquiz, ecatepec, olimpica] .  
64 ?- agregar(algo,[],L).  
L = [algo] .  
65 ?- ■
```

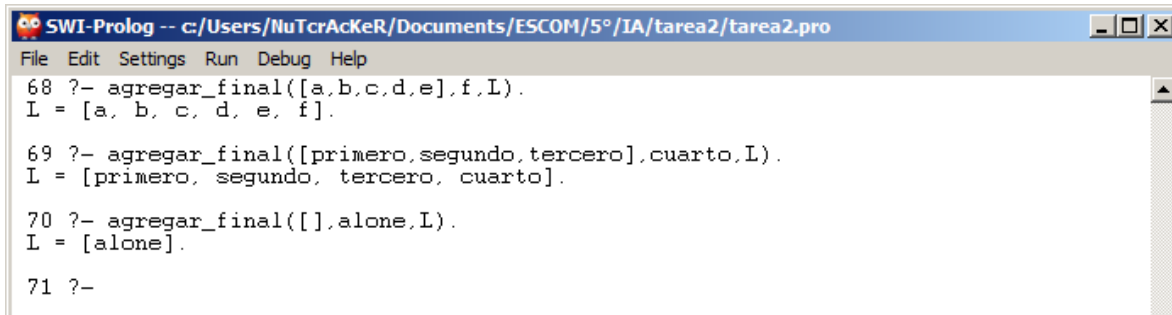
### Ejercicio 3.3

Escriba la(s) regla(s) necesaria(s) para agregar un elemento al final de una lista.

```
?- agregar_final([agua, tierra, viento], fuego, L).  
L = [agua, tierra, viento, fuego]  
true
```

### Solución (3.3)

```
agregar_final([],X,[X|[]]).
agregar_final([H1|T1],X,[H1|T1]):-agregar_final(T1,X,T).
```



```
SWI-Prolog -- c:/Users/NuTcrAcKeR/Documents/ESCOM/5º/IA/tarea2/tarea2.pro
File Edit Settings Run Debug Help
68 ?- agregar_final([a,b,c,d,e],f,L).
L = [a, b, c, d, e, f].

69 ?- agregar_final([primero,segundo,tercero],cuarto,L).
L = [primero, segundo, tercero, cuarto].

70 ?- agregar_final([],alone,L).
L = [alone].

71 ?-
```

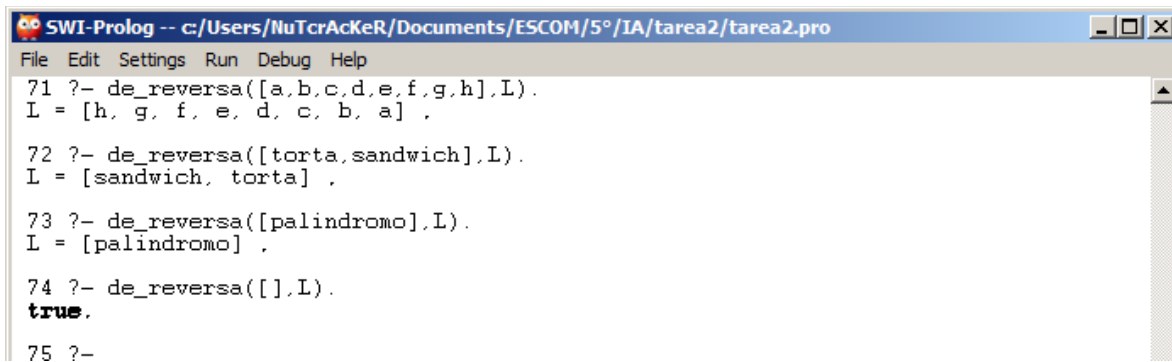
### Ejercicio 3.4.

Escriba la(s) regla(s) necesaria(s) para invertir los elementos de una lista.

```
?- de_reversa([angel, beto, sandra, zoyla],L).
L = [zoyla, sandra, beto, angel]
true
```

### Solución (3.4)

```
de_reversa([],_).
de_reversa([H1|T1],L):- agregar_final(M,H1,L),
                        de_reversa(T1,M).
```



```
SWI-Prolog -- c:/Users/NuTcrAcKeR/Documents/ESCOM/5º/IA/tarea2/tarea2.pro
File Edit Settings Run Debug Help
71 ?- de_reversa([a,b,c,d,e,f,g,h],L).
L = [h, g, f, e, d, c, b, a].

72 ?- de_reversa([torta,sandwich],L).
L = [sandwich, torta].

73 ?- de_reversa([palindromo],L).
L = [palindromo].

74 ?- de_reversa([],L).
true.

75 ?-
```

### Ejercicio 3.5.

Escriba la(s) regla(s) necesaria(s) para obtener el último elemento de una lista.

```
?- ultimo([piolin, silvestre, bugss_bunny, porky],X).
X = porky
true
```

### Solución (3.5)

```
ultimo(X,[X]).
ultimo(X,[_|T]):-ultimo(X,T).
```



```

SWI-Prolog -- c:/Users/NuTcrAcKeR/Documents/ESCOM/5º/IA/tarea2/tarea2.pro
File Edit Settings Run Debug Help
76 ?- ultimo([a,b,c,d,e,f],X).
X = f ,

77 ?- ultimo([perro,gato,pajaro],X).
X = pajaro ,

78 ?- ultimo([last],X).
X = last ,

79 ?- ultimo([],X).
false.

80 ?- █

```

### Ejercicio 3.6.

Escriba la(s) regla(s) necesaria(s) para obtener las permutaciones de una lista.

```

?- permutacion([bart,lisa,maggie],L).
L = [bart, lisa, maggie] ;
L = [bart, maggie, lisa] ;
L = [lisa, bart, maggie] ;
L = [lisa, maggie, bart] ;
L = [maggie, bart, lisa] ;
L = [maggie, lisa, bart] ;

```

### Solución (3.6)

```

permutacion([],[]).
permutacion([X|Y],Z):- permutacion(Y,L),
                        agregar(X,L,Z).

```

```

SWI-Prolog -- c:/Users/NuTcrAcKeR/Documents/ESCOM/5º/IA/tarea2/tarea2.pro
File Edit Settings Run Debug Help
80 ?- permutacion([bart,lisa,maggie],L).
L = [bart, lisa, maggie] ;
L = [lisa, bart, maggie] ;
L = [lisa, maggie, bart] ;
L = [bart, maggie, lisa] ;
L = [maggie, bart, lisa] ;
L = [maggie, lisa, bart] ;
false.

81 ?- permutacion([a,b],L).
L = [a, b] ;
L = [b, a] ;
false.

82 ?- permutacion([a,b,c,d],L).
L = [a, b, c, d] ;
L = [b, a, c, d] ;
L = [b, c, a, d] ;
L = [b, c, d, a] ;
L = [a, c, b, d] ;
L = [c, a, b, d] ;
L = [c, b, a, d] ;
L = [c, b, d, a] ;
L = [a, c, d, b] ;
L = [c, a, d, b] ;
L = [c, d, a, b] ;
L = [c, d, b, a] ;
L = [a, b, d, c] ;
L = [b, a, d, c] ;
L = [b, d, a, c] ;
L = [b, d, c, a] ;
L = [a, d, b, c] ;
L = [d, a, b, c] ;
L = [d, b, a, c] ;
L = [d, b, c, a] ;
L = [a, d, c, b] ;
L = [d, a, c, b] ;
L = [d, c, a, b] ;
L = [d, c, b, a] ;
false.

83 ?- permutacion([pelota],L).
L = [pelota] ;
false.

84 ?- permutacion([],L).
L = [] ;

85 ?- █

```

### Ejercicio 3.7.

Escriba la(s) regla(s) necesaria(s) para verificar si una lista esta ordenada.

```
?- ordenada([agua, fuego, tierra, viento]).
true
?- ordenada([agua, tierra, viento, fuego]).
false
```

### Solución (3.7)

```
primero([H|_],H).
ordenada([]).
ordenada([_]).
ordenada([H|T]):- primero(T,X),
                  (H @< X),
                  ordenada(T).
```

### Ejercicio 3.8.

Escriba la(s) regla(s) necesaria(s) para ordenar los elementos de una lista.

```
?- ordenar([stan, cartman, kyle, kenny], L).
L = [cartman, kenny, kyle, stan];
```

### Solución (3.8)

```
ordenar([],_).
ordenar(L,Z):- permutacion(L,Z),
               ordenada(Z).
```

## Conclusiones

Por la realización de este trabajo, se puede establecer que el lenguaje Prolog está orientado a la Inteligencia Artificial, usando la programación lógica. Gracias a su facilidad de programar y su sencilla sintaxis gramatical y numérica, se pueden escribir rápidamente y con pocos errores programas claramente leíbles, además cualquier usuario puede acceder a él si lo desea y sin problemas de entendimiento.

También utiliza pocos comandos en comparación con otros lenguajes de programación.

Por otra parte en este lenguaje al igual que otros, hay que tener en cuenta la asociatividad de los operadores antes de trabajar con él.

Las listas son la única estructura disponible en Prolog y el uso de estas para una programación de complejidad moderada, es de fundamental importancia. Para esto hay que tener en cuenta la naturaleza recurrente tanto de las llamadas como de las listas en este lenguaje.

Además Prolog se puede trabajar en diferentes sistemas operativos, tales como UNIX, WINDOWS, MAC-OS, entre otros.

## Referencias

[http://www.gedlc.ulpgc.es/docencia/lp/documentacion/GB\\_Prolog.pdf](http://www.gedlc.ulpgc.es/docencia/lp/documentacion/GB_Prolog.pdf)

<http://blog.utp.edu.co/alejandropinto/files/2012/04/Pr%C3%A1cticas-de-Prolog-%E2%80%93-Departamento-de-Ciencia-de-La-Computaci%C3%B3n-e-Inteligencia-Artificial-%E2%80%93-Universidad-de-Alicante.pdf>

[http://larmor.nuigalway.ie/~detinko/prolog\\_manual.pdf](http://larmor.nuigalway.ie/~detinko/prolog_manual.pdf)

[http://www.csupomona.edu/~jrfisher/www/prolog\\_tutorial/1.html](http://www.csupomona.edu/~jrfisher/www/prolog_tutorial/1.html)