

1. Ejercicio de la familia simpson





Listas y estructuras


```
1 % persona(Nombre, Sexo, Pareja, Padres)
2 persona(abraham,hombre,mona,[]).
3 persona(mona,mujer,abraham,[]).
4 persona(clancy,hombre,jacqueline,[]).
5 persona(jacqueline,mujer,clancy,[]).
6 persona(herbert,hombre,_,[abraham,mona]).
7 persona(homero,hombre,marge,[abraham,mona]).
8 persona(marge,mujer,homero,[clancy,jacqueline]).
9 persona(patty,mujer,_,[clancy,jacqueline]).
10 persona(selma,mujer,_,[clancy,jacqueline]).
11 persona(bart,hombre,_,[homero,marge]).
12 persona(lisa,mujer,_,[homero,marge]).
13 persona(maggie,mujer,_,[homero,marge]).
14 persona(ling,mujer,_,[selma]).
15
```

REGLAS

```
16 % reglas
17 hombre(X):-persona(X,hombre,_,_).
18 mujer(X):-persona(X,mujer,_,_).
19 es_progenitor(P,H):-persona(H,_,_,L),member(P,L).
20 es_padre(P,H):-es_progenitor(P,H),hombre(P).
21 es_madre(M,H):-es_progenitor(M,H),mujer(M).
22 es_hijo(H,P):-es_progenitor(P,H),hombre(H).
23 es_hija(H,P):-es_progenitor(P,H),mujer(H).
24
25 es_hermano(H,X):-persona(X,_,_,L),member(P,L),es_progenitor(P,H),X\==H,hombre(H).
26 es_hermana(H,X):-persona(X,_,_,L),member(P,L),es_progenitor(P,H),X\==H,mujer(H).
27
28 es_abuelo(A,N):-es_progenitor(A,X),es_progenitor(X,N),hombre(A).
29 es_abuela(A,N):-es_progenitor(A,X),es_progenitor(X,N),mujer(A).
30
31 es_tio(T,S):-es_progenitor(P,S),es_hermano(T,P),hombre(T).
32 es_tia(T,S):-es_progenitor(P,S),es_hermana(T,P),mujer(T).
33
34 es_sobrino(S,T):-es_progenitor(P,S),(es_hermano(P,T);es_hermana(P,T)),hombre(S).
35 es_sobrina(S,T):-es_progenitor(P,S),(es_hermano(P,T);es_hermana(P,T)),mujer(S).
36
37
38 es_nieto(N,A):-es_progenitor(A,X),es_progenitor(X,N),hombre(N).
39 es_nieta(N,A):-es_progenitor(A,X),es_progenitor(X,N),mujer(N).
40
```


DEMOSTRACIONES

 <code>es_padre(Padre,homero)</code> Padre = abraham false	 <code>es_padre(Padre,marge)</code> Padre = clancy false
?- <code>es_padre(Padre,homero)</code>	?- <code>es_padre(Padre,marge)</code>
 <code>es_madre(Madre,patty)</code> Madre = jacqueline	 <code>es_madre(Madre,bart)</code> Madre = marge
?- <code>es_madre(Madre,patty)</code>	?- <code>es_madre(Madre,bart)</code>

 `es_esposo(Esposo,marge)`

Esposo = homero

?- `es_esposo(Esposo,marge)`

 `es_esposo(Esposo,mona)`


Esposo = abraham

?- `es_esposo(Esposo,mona)`

 `es_esposa(Esposa,clancy)`

Esposa = jacqueline

?- `es_esposa(Esposa,clancy)`

 `es_esposa(Esposa,homero)`

Esposa = marge

?- `es_esposa(Esposa,homero)`

 `es_hijo(Hijo,homero)`

Hijo = bart

Next | 10 | 100 | 1,000 | Stop

?- `es_hijo(Hijo,homero)`

 `es_hijo(Hijo,abraham)`

Hijo = homero

Hijo = herbert

?- `es_hijo(Hijo,abraham)`


 `es_hija(Hija,clancy)`

Hija = marge

Hija = selma

Hija = patty



?- `es_hija(Hija,clancy)`

 `es_hija(Hija,homero)`

Hija = lisa

Hija = magie

?- `es_hija(Hija,homero)`

 <code>es_hermano(Hermano,homero)</code> Hermano = herbert Next 10 100 1,000 Stop <hr/> ?- <code>es_hermano(Hermano,homero)</code> <hr/> Hermana = selma Hermana = patty <hr/> ?- <code>es_hermana(Hermana,marge)</code>	 <code>es_hermana(Hermana,bart)</code> Hermana = lisa Hermana = magie Next 10 100 1,000 Stop <hr/> ?- <code>es_hermana(Hermana,bart)</code> <hr/> Hermana = magie <hr/> ?- <code>es_hermana(Hermana,lisa)</code>
--	--

Sobrino = bart false <hr/> ?- <code>es_sobrino(Sobrino,herbert)</code> <hr/>	Sobrino = bart false  <hr/> ?- <code>es_sobrino(Sobrino,selma)</code> <hr/>
Sobrina = lisa Sobrina = lisa Sobrina = magie Sobrina = magie false <hr/> ?- <code>es_sobrina(Sobrina,herbert)</code> <hr/>	 <code>es_sobrina(Sobrina,patty)</code> Sobrina = lisa Sobrina = lisa Sobrina = magie Sobrina = magie Sobrina = ling Sobrina = ling <hr/> ?- <code>es_sobrina(Sobrina,patty)</code> <hr/>

Tio = herbert false	Tio = herbert false
?- es_tio(Tio,bart)	?- es_tio(Tio,magie)
Tia = selma Tia = patty false	Tia = marge Tia = patty false
?- es_tia(Tia,bart)	?- es_tia(Tia,ling)

es_abuelo(Abuelo,bart) Abuelo = abraham Abuelo = clancy false	es_abuelo(Abuelo,lisa) Abuelo = abraham Abuelo = clancy false
?- es_abuelo(Abuelo,bart)	?- es_abuelo(Abuelo,lisa)
Abuela = mona Abuela = jacqueline false	Abuela = jacqueline false
?- es_abuela(Abuela,lisa)	?- es_abuela(Abuela,ling)

Nieto = bart false	Nieto = bart false
?- es_nieto(Nieto,clancy)	?- es_nieto(Nieto,abraham)
⚙ es_nieta(Nieta,clancy)	⚙ es_nieta(Nieta,abraham)
Nieta = lisa	Nieta = lisa
Nieta = magie	Nieta = magie
Nieta = ling	alse
?- es_nieta(Nieta,clancy)	?- es_nieta(Nieta,abraham)

2. Ejercicio de rutas entre ciudades de Canada

Listas y estructuras

```

1 % =====
2 % RUTAS ENTRE CIUDADES DE CANADÁ
3 % =====
4
5 ciudad(vancouver, [conexion(edmonton,16), conexion(calgary,13)]).
5 ciudad(edmonton, [conexion(saskatoon,12), conexion(calgary,4)]).
7 ciudad(calgary, [conexion(regina,14), conexion(edmonton,4)]).
3 ciudad(saskatoon, [conexion(calgary,9), conexion(winnipeg,20)]).
9 ciudad(regina, [conexion(winnipeg,4), conexion(saskatoon,7)]).
3 ciudad(winnipeg, []).
1

```


REGLAS

```
%reglas
existe_conexion(O,D):- ciudad(O,L),member(conexion(D,_),L).
valor(O,D,C):- ciudad(O,L),member(conexion(D,C),L).
existe_arista(A,B):- existe_conexion(A,B);existe_conexion(B,A).
costo_intermedio(A,C,T):- valor(A,B,C1),valor(B,C,C2),T is C1+C2.
es_posible_viajar(A,C):- existe_conexion(A,B),existe_conexion(B,C).
costo_viaje(O,D,C):- valor(O,D,C).
costo_viaje(O,D,C):- valor(O,X,C1),costo_viaje(X,D,C2),C is C1+C2.
```

DEMOSTRACIONES

```
⚙ existe_conexión(saskatoon, Destino)
Destino = calgary
Destino = winnipeg
?- existe_conexión(saskatoon, Destino)
```


 `valor(calgary, Destino, Costo).`

Costo = 14,

Destino = regina

Costo = 4,

Destino = edmonton

?- `valor(calgary, Destino, Costo).`

 `existe_arista(calgary, Existe)`


Existe = regina

Existe = edmonton


Existe = vancouver

Existe = saskatoon

?- `existe_arista(calgary, Existe)`

 `es_posible_viajar(edmonton, _, calgary).`

true

 `es_posible_viajar(edmonton, T, calgary).`

T = saskatoon

?- `es_posible_viajar(edmonton, T, calgary).`

Conclusiones:

El ejercicio permitió comprender cómo las estructuras y listas en Prolog facilitan la organización de información compleja, como las relaciones familiares. Representar la familia Simpson mediante hechos estructurados hizo que las consultas fueran más claras y el código más flexible. Además, se reforzó el razonamiento lógico al aplicar reglas para deducir parentescos sin usar programación imperativa, lo que demuestra la potencia del paradigma lógico.

En el ejercicio de las ciudades, se aplicaron estructuras y listas para modelar conexiones y rutas entre distintas localidades. Esto permitió calcular costos de viaje y verificar posibles trayectos de manera más eficiente. La experiencia ayudó a entender cómo la recursividad y el uso de listas hacen posible resolver problemas reales de rutas y caminos, reforzando la lógica de búsqueda y el pensamiento estructurado en Prolog.

ProgIIIIG101-Act05-

Santiago Henao

-Johany Ballesteros

-Juan Guillermo Galindo