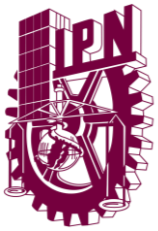

INSTITUTO POLITECNICO NACIONAL

ESCUELA SUPERIOR DE COMPUTO

INGENIERIA EN INTELIGENCIA ARTIFICIAL

PARADIGMAS DE PROGRAMACION

PRACTICA NUMERO 5

TECNICAS DE PROGRAMACION LOGICA

VAZQUEZ PEREZ ANGEL MARTIN

GRUPO 3BV2

INTRODUCCION

En esta práctica estaremos platicando métodos que usamos en la práctica pasada la cual trato sobre la lógica de primer orden.

En este caso en particular estaremos ya implementando recursividad y bases de datos relacionales para que prolog analice y nos retorne el valor que nosotros requeriremos y necesitamos según lo que se solicite. Quizá necesitemos realizar algunas modificaciones a nuestra base para crear más relaciones ya que lógicamente prolog no nos proporciona realizar relaciones entre entidades o tablas.

La recursividad que estará implementada en la práctica será una recursividad directa, se estará mandando a llamar a sí misma hasta que se llegue al caso base, este caso base ya son los que están definidos en las bases de conocimientos. Estos casos base son los que conocemos en el análisis y diseño de algoritmos recursivos que siempre deben de tener un caso base el cual nos va a indicar que la función nos va a retornar ese valor cuando llegue a ese caso.

Dentro de estas funciones igualmente veremos la sucesión de Fibonacci la cual es una sucesión de números que se genera con la suma de los anteriores 2 números del número, igualmente mostraremos la factorial. Esta se hará mediante funciones recursivas.

Igualmente se nos estar presentando un problema de lógica de esos que te hacen pensar y tenemos nosotros que enseñarle a prolog a crear relaciones entre la información que se nos esté presentando para así que prolog nos devuelva el resultado del a pregunta que se nos presente en la práctica.

Igualmente tendremos que implementar la teoría del generar comprobar que nos va a servir para verificar varias reglas que podremos nosotros mismos crear u con esas reglas ingresarle valores u que por medio de esta técnica que nos devuelva si es cierto o si es falso.

1. Aplicando la técnica generar – comprobar, resuelva los siguientes problemas planteados.

1.1 Una comida consiste en una entrada, un plato principal y un postre. Cree una base de conocimientos con distintos tipos de comida y sus valores calóricos, Deberá producir un menú con comida light (valor calórico menor de 10Kcal) para personas con problemas de sobre peso, uno entre 10Kcal y 20Kcal para personas razonables, y otro mayor de 20Kcal para glotones de verdad.

BASE DE CONOCIMIENTOS

```
entrada(sopa_de_verduras,1800).
entrada(arroz,2800).
entrada(sopa_de_tornillo,3200).
entrada(crema_de_zanahoria,6000).
plato_fuerte(pechuga_asada,4000).
plato_fuerte(chilaquiles,7500).
plato_fuerte(chuleta_de_cerdo,5000
plato_fuerte(lasagna,12000).
postre(roles_de_canela,5000).
postre(pure_de_manzana,1500).
postre(crepa,4500).
postre(concha_rellena,7500).
```

FUNCIONES

```
menoscalorias(X,Y,Z):- 10000> (X+Y+Z). %comprobar
mediacalorias(X,Y,Z):- 10000< (X+Y+Z) , (X+Y+Z) < 20000. %comprobar
muchascalorias(X,Y,Z):- 20000< (X+Y+Z). %comprobar
```

```
menulight(Entrada, Plato, Postre) :-entrada(Entrada, N),plato_fuerte(Plato, M),postre(Postre, J),
    menoscalorias(N, M, J), %comprobar
    SumaCalorias is N + M + J, %generar
```

```
menumedio(Entrada, Plato, Postre) :-entrada(Entrada, N),plato_fuerte(Plato, M),postre(Postre, J),
    mediacalorias(N, M, J), %comprobar
    SumaCalorias is N + M + J,%generar
```

```
menugloton(Entrada, Plato, Postre) :-entrada(Entrada, N),plato_fuerte(Plato, M),postre(Postre, J),
    muchascalorias(N, M, J), %comprobar
    SumaCalorias is N + M + J,%generar
```

Analizando las funciones antes mencionadas tuvimos que implementar unas funciones que nos indicen las calorías, si son aptas para el menú light, o para el menú medio o para el menú de glotón, y estas funciones las ocuparemos en la función que nos va a generar los menús, dándonos como resultado lo siguiente.

Para el menú light, nos generó 5 menús, 2 de estos son los siguientes, corroboramos que los menús no sobrepasen las 10 mil calorías, y es correcto.

```
?- menulight(Entrada, Plato, Postre).
Menu Light:
Entrada: sopa_de_verduras : 1800
Plato Fuerte: pechuga_asada : 4000
Postre: pure_de_manzana : 1500
Total de calorias: 7300
Entrada = sopa_de_verduras,
Plato = pechuga_asada,
Postre = pure_de_manzana ;
Menu Light:
Entrada: sopa_de_verduras : 1800
Plato Fuerte: chuleta_de_cerdo : 5000
Postre: pure_de_manzana : 1500
Total de calorias: 8300
Entrada = sopa_de_verduras,
Plato = chuleta_de_cerdo,
Postre = pure_de_manzana ;
Menu Light:
Entrada: arroz : 2800
Plato Fuerte: pechuga_asada : 4000
Postre: pure_de_manzana : 1500
Total de calorias: 8300
```

Ahora le solicitamos un menú glotón:

```
?- menugloton(Entrada, Plato, Postre).
Menu Gloton:
Entrada: sopa_de_verduras : 1800
Plato Fuerte: lasagna : 12000
Postre: concha_rellena : 7500
Total de calorias: 21300
Entrada = sopa_de_verduras,
Plato = lasagna,
Postre = concha_rellena ;
Menu Gloton:
Entrada: arroz : 2800
Plato Fuerte: lasagna : 12000
Postre: concha_rellena : 7500
Total de calorias: 22300
```

Ahora menú promedio

```
?- menumedio(Entrada, Plato, Postre).
Menu Medio:
Entrada: sopa_de_verduras : 1800
Plato Fuerte: pechuga_asada : 4000
Postre: roles_de_canela : 5000
Total de calorias: 10800
Entrada = sopa_de_verduras,
Plato = pechuga_asada,
Postre = roles_de_canela ;
Menu Medio:
Entrada: sopa_de_verduras : 1800
Plato Fuerte: pechuga_asada : 4000
Postre: crepa : 4500
Total de calorias: 10300
Entrada = sopa_de_verduras,
Plato = pechuga_asada,
Postre = crepa ;
```

EJERCICIO 1.2

Este pasatiempo consta de las siguientes afirmaciones:

- **Un alemán, un británico y un sueco viven cada uno en una casa de color diferente, tienen diferentes mascotas y gustan diferentes bebidas**
  - **El alemán vive en la casa verde**
  - **El sueco bebe café**
  - **Al británico no le gustan los gatos**
  - **Hay un perro en la casa blanca**
  - **El sueco vive junto a la casa azul**
  - **Alguien bebe agua y tiene un pez**

Se plantea la siguiente pregunta:

- **¿Quién bebe té?**

Este problema fue muy complejo ya que tuvimos que encontrar las relaciones entre cada persona junto a sus casas y asus bebidas e incluso escribir en la base de conocimientos la información que nos proporcionaban,

Primeramente se declararon los hechos, donde se escribían todos los objetos que se nos mencionan en el problema, después las relaciones que estos nos daban .

```
color(azul).
color(blanco).
color(verde).
mascota(perro).
mascota(pez).
mascota(gato).
bebida(cafe).
bebida(te).
bebida(agua).
pais(sueco).
pais(britanico).
pais(aleman).

relacion_casa_mascota(blanco,perro).%bc
relacion_mascota_bebida(pez,agua).%cd
relacion_pais_casa(aleman,verde).%ab
relacion_bebida_pais(cafe,sueco).%da
nohayrelacion_pais_casa(britanico,blanco).
nohayrelacion_pais_casa(sueco,azul).
norelacionact(britanico,gato).
```

Para continuar se tuvieron que crear varias funciones para que nos devolviera la persona que cumplía con ese criterio, esas funciones iban a comprobar y devolver la persona con la que encontrar la relacion, y concatenando todas las funciones elaboradas en el código se llama a la sentencia que usaremos en este caso nosotros le podemos enviar cualquier consulta y prolog se encargara de crear todas las relaciones y ver cual nos devuelve TRUE y dentro de todas las comparaciones nos devolverá el objetivo que queremos

```
humano(X,Y,Z,W):-pais(X),color(Y),mascota(Z),bebida(W),
    relacionpaiscasa(X,Y),casamascotarelacion(Y,Z),mascotabebidarelacion(Z,W),
    bebidapaisrelacion(W,X),mascotapaisrelacion(X,Z).
```

PASAmos la consulta a prolog y nos devuelve que es el alemán que vive en la casa verde y este también tiene un gato,aunque suene un poco extraño que el alemán beba te, analizando con lógica y preguntando a varios compañeros sobre el acertijo todos llegamos a la misma conclusión y prolog nos devulve este resultado

```
?- humano(X,Y,Z,te).
X = aleman,
Y = verde,
Z = gato ;

?-
```

2. Aplicando la técnica de recursividad resuelva los problemas planteados.

Todo el código solicitado en este punto deberá ir en un solo archivo a menos que se indique lo contrario.

2.1 Obtenga la factorial de un número.

En esta función nuestro caso base es la factorial de 0 que es 1, y dentro de la regla de factorial se debe de introducir, el número que queremos conocer su factorial el cual es X, y Facto será nuestro resultado. Dentro de la función se decrementa 1 a nuestro valor que queremos conocer y llamamos recursivamente a la función con este valor, y otra variable para conocer la factorial de ese, al final se multiplican estos.

```
factorial(0, 1). % Caso base

factorial(X, Facto) :-
    Y is X-1,
    factorial(Y,Factorialdelfactorial),
    Facto is X * Factorialdelfactorial.
```

```
?- factorial(6,X).
X = 720 .

?- factorial(7,X).
X = 5040 .

?- factorial(5,X).
X = 120 .
```

2.2 Obtenga la sucesión de Fibonacci.

Tenemos 3 casos base dentro de este, que son Fibonacci de 0 es 0, Fibonacci de 1 es 1 y Fibonacci de 2 números es 1.

Dentro de la función Fibonacci se escribe la cantidad de números que queremos generar con la sucesión de Fibonacci, se comprueba si n es mayor de 2 para continuar, después se genera Y y Z los cuales entrarían a la misma función recursiva de Fibonacci, para después X sea la suma de los Fibonacci de estos 2 números. Y la función de generarfibonaci se introduce el inicio y el fin de nuestra generación de Fibonacci. Y se escribe ese valor siguiendo de una coma y después se genera recursivamente el que sigue.

```
fibonacci(0, 0).
fibonacci(1, 1).
fibonacci(2, 1).

fibonacci(N, X) :-
    N > 2, %comprobar
    Y is N - 1,%generar
    Z is N - 2,%generar
    fibonacci(Y, Fi1),%generar
    fibonacci(Z, Fi2),%generar
    X is Fi1 + Fi2.

generarfibonacci(Ini, Fin) :-
    Ini =< Fin, %comprobar
    fibonacci(Ini, X), %generar
    write(X), write(', '),
    Sig is Ini + 1,
    generarfibonacci(Sig, Fin). %generar
```

```
?- generarfibonacci(3,8).
2, 3, 5, 8, 13, 21,
false.

?- generarfibonacci(0,9).
0, 1, 1, 2, 3, 5, 8, 13, 21, 34,
false.

?- generarfibonacci(7,12).
13, 21, 34, 55, 89, 144,
false.
```

2.5 El algoritmo de Euclides que calcula el máximo común divisor de dos números.

Acá nuestro caso base es cuando el resultado y el mismo valor es 0, entonces le indicamos a porlog que se detenga y ya no siga calculando entonces se va a devolver este X el cual será el resultado de nuestra MCD.

Dentro de la función de Euclides se verifica que el número que se mandara como residuo sea mayor a 9, si es igual a 0 ya se detiene así, después el residuo se calcula con el módulo, ya que así trabaja este algoritmo.

```
euclidesMcd(X, 0, X) :- !.
euclidesMcd(X, Y, Resultado) :-
    Y > 0,
    Residuo is (X mod Y),
    euclidesMcd(Y, Residuo, Resultado).
```

```
?- euclidesMcd(48,18,X).
X = 6.

?- euclidesMcd(108,12,X).
X = 12.

?- euclidesMcd(121,55,X).
X = 11.

?- euclidesMcd(300,70,X).
X = 10.
```

3. Aplicando la técnica de relaciones y bases de datos relacionales resuelva los problemas planteados.

3.1 Dadas las siguientes tablas cree la base de conocimientos de prolog y genere las reglas de consultas vistas en clase para lograr el algebra relacional, exponga todos los resultados
según sea el caso y conteste las preguntas (consultas).
A) ¿Quién fue el departamento que más vendió?

Ocuparemos varias funciones, dentro de la del departamento más vendido hay que conocer cuanto vendió cada departamento y después introducir esta cantidad a la función cual es más meterá estos valores a una lista y conocemos el Máximo y se devuelve este valor

```
cualesmas(Dep,X,Y,Z,W,M,Maximo):-
    max_list([X,Y,Z,W,M], Maximo),
        (
        (Maximo = X, Dep = d1);
        (Maximo = Y, Dep = d2);
        (Maximo = Z, Dep = d3);
        (Maximo = W, Dep = d4);
        (Maximo = M, Dep = d5)).

departamentomasvendio(Dep):-
    tienetres(d1,A,B),  vendio(A,AP),vendio(B,BP), D1 is AP+BP,
    tiene(d2,C,D,E),vendio(C,CP),vendio(D,DP),vendio(E,EP), D2 is CP+DP+EP,
    tienetres(d3,F,G), vendio(F,FP),vendio(G,GP),  D3 is FP+GP,
    tiene(d4,H,I,J),vendio(H,HP),vendio(I,IP), vendio(J,JP), D4 is HP+IP+JP,
    tiene(d5,K,L,M), vendio(K,KP),vendio(L,LP),vendio(M,MP), D5 is KP+LP+MP,
    cualesmas(Dep,D1,D2,D3,D4,D5,Maximo),
    write('el departamento '), write(Dep), write(' vendio esta cantidad'),write(Maximo).
```

```
?- departamentomasvendio(Dep).
el departamento d1 vendio esta cantidad177638.0
Dep = d1 ;
false.
```

B) ¿Qué cantidad se vendió de cada producto?

Para la cantidad de producto por medio de las relaciones dividíamos total entre precio y lo mostramos a pantalla.

```
cantidadvproducto(Codigo, Cantidad) :-
    producto(Codigo, _, Precio, _),
    vendio(Codigo, Total),
    Cantidad is Total / Precio,
    write('el producto '), write(Codigo),write(' vendio '),write(Cantidad),write(' productos').
```

```
?- cantidadvproducto(X,Y).
el producto 102 vendio 3.0 productos
X = 102,
Y = 3.0 ;
el producto 104 vendio 71.0 productos
X = 104,
Y = 71.0 ;
el producto 110 vendio 37.0 productos
X = 110,
Y = 37.0 ;
el producto 112 vendio 7.0 productos
X = 112,
Y = 7.0 ;
el producto 113 vendio 3.0 productos
X = 113,
Y = 3.0 ;
```

A continuación se muestra en la tabla la cantidad de productos que se vendieron

ID PRODUCTO	CANTIDAD VENDIDA
102	3
104	71
110	37
112	7
113	3
120	8
121	5
122	4
125	37
128	3
130	3
134	2
137	5

C) ¿Quién es el vendedor del mes?

La función de vendedor del mes es igualita a la de cual departamento vende solo se hace al final una relación entre el vendedor y el departamento y se muestra.

```
vendedordelmes(Vendedor):-
    tienetres(d1,A,B),  vendio(A,AP),vendio(B,BP), D1 is AP+BP,
    tiene(d2,C,D,E),vendio(C,CP),vendio(D,DP),vendio(E,EP), D2 is CP+DP+EP,
    tienetres(d3,F,G), vendio(F,FP),vendio(G,GP),  D3 is FP+GP,
    tiene(d4,H,I,J),vendio(H,HP),vendio(I,IP), vendio(J,JP), D4 is HP+IP+JP,
    tiene(d5,K,L,M), vendio(K,KP),vendio(L,LP),vendio(M,MP), D5 is KP+LP+MP,
    cualesmas(Dep,D1,D2,D3,D4,D5,Maximo),
    vendedor(_,Vendedor,Dep),
    write('el vendedor del mes es '), write(Vendedor), write(' vendio esta cantidad'),write(Maximo).
```

```
?- vendedordelmes(X).
el vendedor del mes es paulo vendio esta cantidad177638.0
X = paulo ;
false.
```

D) Si por cada venta realizada les pagan el 3 % en comisión, ¿Cuánto gano cada vendedor?

Para esta fue de las mas complicadas ya que tuvimos que crear varias funciones donde se conectaba cada una con la otra

```
comision(Vendedor, ComisionTotal) :-
vendedor(_, Vendedor, Dep),
departamento(Dep, _),
comision_aux(Dep, ComisionTotal).

comision_aux(Dep, ComisionTotal) :-
tienetres(Dep, A, B),
comision_producto(A, ComA),
comision_producto(B, ComB),
ComisionTotal is ComA + ComB.

comision_aux(Dep, ComisionTotal) :-
tiene(Dep, A, B, C),
comision_producto(A, ComA),
comision_producto(B, ComB),
comision_producto(C, ComC),
ComisionTotal is ComA + ComB + ComC.

comision_producto(Codigo, Comision) :-
producto(Codigo, _, Precio, _),
vendio(Codigo, Total),
Comision is (Total / Precio) * (Precio * 0.03).

imprimir_comisiones :-
vendedor(Codigo, Nombre, _),
comision(Nombre, ComisionTotal),
write('El vendedor '), write(Nombre), write(' se llevo de comision: '), write(ComisionTotal), nl.
```

Primeramente tenemos la que se llama comisión que me va a devolver la comisión que el vendedor tuvo , y esto apoyándose de otra función donde nos va a regresar la suma de las comisiones de cada producto que se tenga, se saca la cantidad que vendioe el vendedor, para luego s emultipique por .3 y el valor del producto, para que luego se sumen todos los productos del mismo departamento y así nos regrese la comisión

```
?- imprimir_comisiones.
El vendedor paulo se llevo de comision: 5329.139999999999
true ;
El vendedor virginia se llevo de comision: 2880.0
true ;
El vendedor maria se llevo de comision: 4482.0
true ;
El vendedor mauricio se llevo de comision: 2798.25
true ;
El vendedor patricia se llevo de comision: 3038.2799999999997
true.
```

E) Respalde la respuesta de (A), buscando el valor mayor de ganancia de cada vendedor.

Podemos observar que obtuvimos como resultado que el departamento numero 1 fue el que vendio mas, en el inciso A, mientras que en el inciso D el vendedor que mas comisión tuvo fue PAULO el cual justo trabaja en ese departamento, entonces queda demostrado que nuestras 2 funciones están correctas .

CONCLUSION

Pudimos finalizar esta practica de prolog con muchas dificultades, después de mucho tiempo analizando como poder resolver los problemas, llegamos a la conclusión de que es muy complicado este lenguaje ,ya que todo es de manera lógica, y muy difícil crear funciones y que nos regresen un resultado que nosotros querramos , para este volver a ocuparlo Entonces hay que saber como relacionar todas las funciones que podamos implementar.

Igual supimos que hay que tener mucho cuidado con la recursividad ya que nos puede ocasionar que la pila se nos este llenando la que tenemos nosotros en nuestra memoria RAM y esto ocasione que nos la acabemos, por lo que prolog al detetar esto finaliza el programa.

Igualmente en la base de datos relacionales no nos otorgan todas las relaciones que nosotros necesitamos para poder crear las funciones que se nos solicitaban en la ptactica pero pudimos salir.