



VERACRUZ
GOBIERNO
DEL ESTADO



SEV
Secretaría
de Educación



DET
Dirección de Educación
Tecnológica del Estado
de Veracruz



Instituto Tecnológico Superior de Chicontepec

Ingeniería en Sistemas Computacionales

Nombre:

Camelia Bautista Hernández

Docente:

Ing. Efrén Flores Cruz

Asignatura:

Programación Lógica y Funcional

Unidad 3

Programación Lógica

Trabajo:

Ejercicios en SWI-Prolog.

8° SEMESTRE



VERACRUZ
GOBIERNO
DEL ESTADO



SEV
Secretaría
de Educación



DET
Dirección de Educación
Tecnológica del Estado
de Veracruz



Contenido

INTRODUCCION	3
DESARROLLO	4
FUNCIONAMIENTO CON LOS ARCHIVOS .PL	4
Ejercicios de alumnos de la carrera de isc8	6
EJERCICIO DE MUNICIPIOS.....	7
DATOS DEL TECLADO	9
ÁRBOL GENEALÓGICO	11
MULTIPLICACIÓN DE 2 NÚMEROS	13
EDAD ACTUAL	14
FACTORIAL DE UN NUMERO.....	16
NUMERO PAR	17
NÚMERO IMPAR	19
CONCLUSIÓN	21

INTRODUCCION

En la siguiente actividad en los ejercicios en SWI-Prolog El lenguaje de programación Prolog es el lenguaje más conocido de programación lógica. La programación lógica es un tipo de programación declarativa que nos permite modelar nuestro problema en base a una serie de hechos y predicados que, aplicados a la entrada del usuario, permiten al intérprete inferir automáticamente la solución.

SWI-Prolog es una implementación en código abierto del lenguaje de programación Prolog. Su autor principal es Jan Wielemaker. En desarrollo ininterrumpido desde 1987, SWI-Prolog posee un rico conjunto de características, bibliotecas (incluyendo su propia biblioteca para GUI, XPCE), herramientas (incluyendo un IDE) y una documentación extensiva. SWI-Prolog funciona en las plataformas Unix, Windows y Macintosh.

El nombre SWI deriva de Sociaal-Wetenschappelijke Informatica ("Informática de Ciencias Sociales"). El nombre de ese grupo se cambió posteriormente a HCS (Human-Computer Studies).

Esta actividad consiste en la resolución de algunos ejercicios usando un archivo con extensión de .PL, aquí está ubicado la programación de que se desea resolver. Posteriormente este archivo se abre en SWI-Prolog de los ejercicios que se muestran.

DESARROLLO

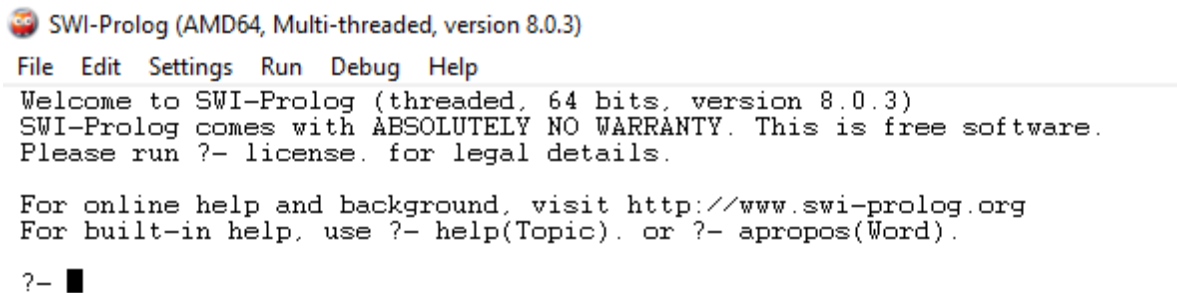
FUNCIONAMIENTO CON LOS ARCHIVOS .PL

Los programas Prolog hay que editarlos con un editor de texto. Se puede usar el Notepad, editor, pero no conviene, es bastante limitado ya que en este caso se realiza en el SWI-Prolog.

Hay uno que se llama emacs que viene con el SWI. También se pueden usar editores de texto de uso general, p.ej. el UltraEdit. En la página de editores contamos algunos tipos sobre cómo trabajar con los editores. Conviene guardar los archivos con extensión .pl.

Primero se abre el programa de SWI-Prolog

En la siguiente imagen se muestra el área de trabajo de SWI-Prolog.

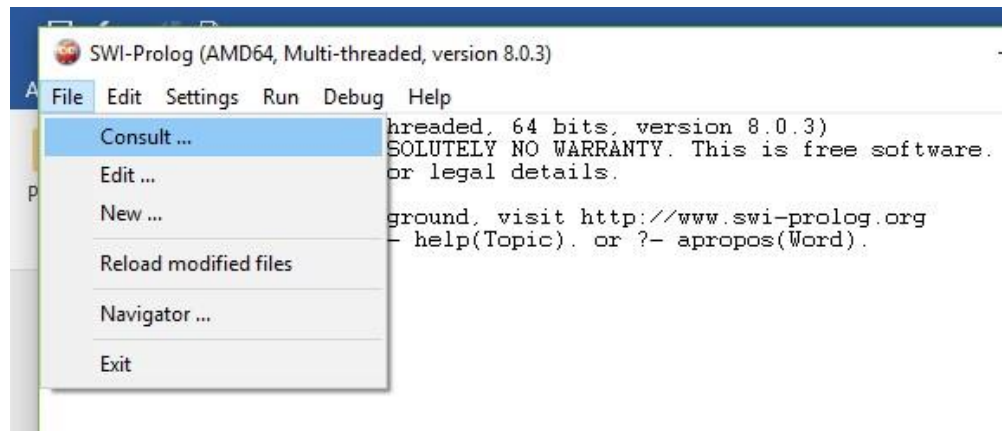


```
SWI-Prolog (AMD64, Multi-threaded, version 8.0.3)
File Edit Settings Run Debug Help
Welcome to SWI-Prolog (threaded, 64 bits, version 8.0.3)
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software.
Please run ?- license. for legal details.

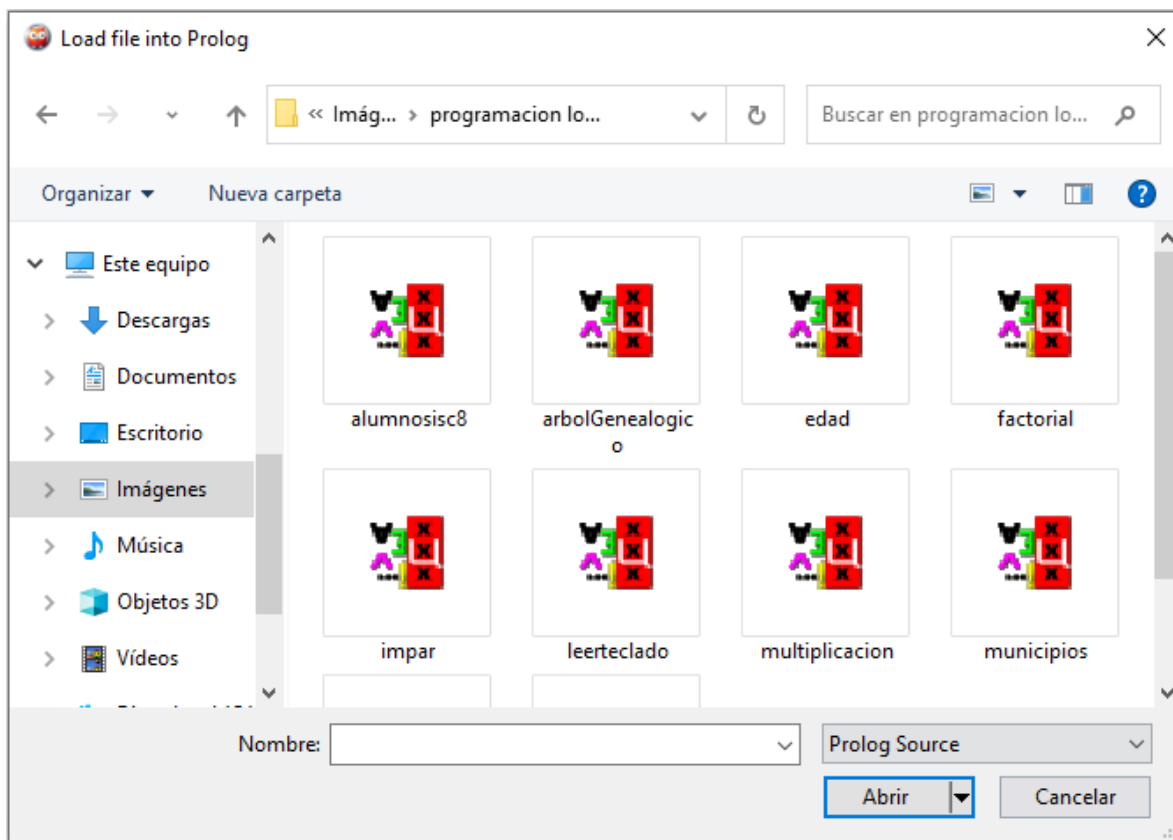
For online help and background, visit http://www.swi-prolog.org
For built-in help, use ?- help(Topic). or ?- apropos(Word).

?- █
```

Para la realización de los siguientes ejercicios a continuación se va a realizar en un editor de SWI-Prolog la cual descargamos, y se guarda el archivo se va a guardar con la extensión de .PL; una vez finalizada la programación en el software de SWI-Prolog, se va a la pestaña de File en la primera opción que dice Consult.

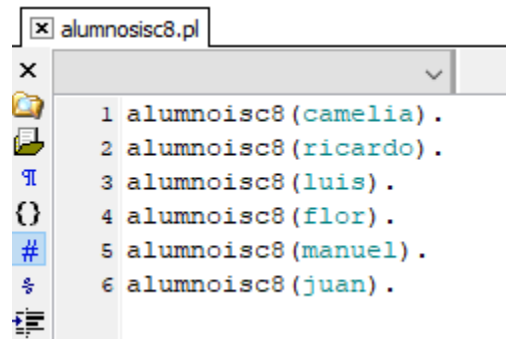


Seleccionamos el archivo que se deseamos ejecutar y abrir el archivo .



Ejercicios de alumnos de la carrera de isc8

Se realiza la programación en algún editor de texto de en SWI-Prolog editor, este ejercicio nos muestra todos los alumnos de la carrera de sistemas de octavo semestre.



```
1 alumnoisc8(camelia).  
2 alumnoisc8(ricardo).  
3 alumnoisc8(luis).  
4 alumnoisc8(flor).  
5 alumnoisc8(manuel).  
6 alumnoisc8(juan).
```

Seleccionamos el archivo de alumnosisc8, ingresamos los datos del alumno que se desea saber si es nos muestra en el programa ejecutado como por ejemplo si la alumna camelia se encuentra en el grupo con la siguiente instrucción: `alumnoisc8(camelia)`. Como si corresponde a este grupo arroja el mensaje de `true`, es decir que es verdad, si se agrega la misma instrucción, pero ahora con el nombre de alguna persona que no es de este grupo por ejemplo pedro, la respuesta es `false` (falso).

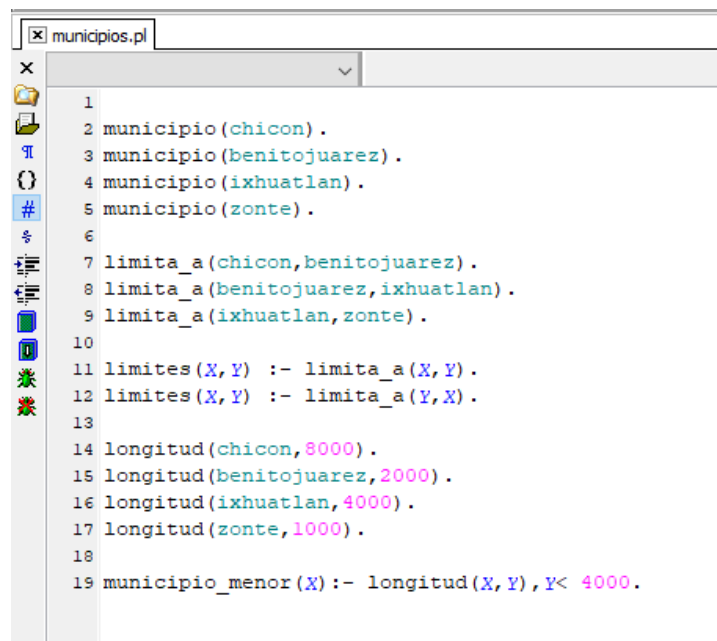
```
?-  
% d:/RCamelia/Pictures/programacion logica y funcional/alumnosisc8.pl compiled 0.00 sec, 6 clauses  
?- alumnoisc8(camelia).  
true.  
  
?- alumnoisc8(luis).  
true.  
  
?- alumnoisc8(angel).  
false.  
  
?- alumnoisc8(lulu).  
false.
```

A continuación nos muestra los nombres de los integrantes de los alumnos de la carrera de ingeniería en sistemas computacionales las cuales son 6 alumnos para realizarlo ejecutamos el programa agregamos alumnoisc8(x). una vez ejecutado agregamos ; y nos muestra todos los nombres de los estudiantes de la carrera de ingeniería en sistemas computacionales.

```
% d:/RCamelia/Pictures/programacion logica y funcional/alumnosisc8.pl compiled 0.00 sec, 0 clauses
|
|      alumnoisc8(X).
|      X = camelia ;
|      X = ricardo ;
|      X = luis ;
|      X = flor ;
|      X = manuel ;
|      X = juan.
|
? -
```

EJERCICIO DE MUNICIPIOS

A continuación, en el siguiente ejercicio se realiza sobre los municipios. Se agregan primero los nombres de los municipios, también se agregan los límites de los municipios respectivamente a su territorio con el nombre de los municipios, así como también se agregan las variables de X, Y. por último se agregan los números de los territorios de los municipios. También se agrega una regla para definir los municipios que sean mayores de 4000 en territorio.



```
1
2 municipio(chicon).
3 municipio(benitojuarez).
4 municipio(ixhuatlan).
5 municipio(zonte).
6
7 limita_a(chicon,benitojuarez).
8 limita_a(benitojuarez,ixhuatlan).
9 limita_a(ixhuatlan,zonte).
10
11 limites(X,Y) :- limita_a(X,Y).
12 limites(X,Y) :- limita_a(Y,X).
13
14 longitud(chicon,8000).
15 longitud(benitojuarez,2000).
16 longitud(ixhuatlan,4000).
17 longitud(zonte,1000).
18
19 municipio_menor(X) :- longitud(X,Y), Y < 4000.
```

Para verificar los datos de los límites se escribe el comando de límites (X, Y). Se podrán observar los municipios correspondientes en X, Y con los que se colindan.

```
?-  
% d:/RCamelia/Pictures/programacion logica y funcional/municipios.pl compiled 0.00 sec, 14 clauses  
?- limites(X,Y).  
X = chicon,  
Y = benitojuarez ;  
X = benitojuarez,  
Y = ixhuatlan ;  
X = ixhuatlan,  
Y = zonte ;  
X = benitojuarez,  
Y = chicon ;  
X = ixhuatlan,  
Y = benitojuarez ;  
X = zonte,  
Y = ixhuatlan.  
  
?- ;
```

Los siguientes resultados se refieren a la regla de menor a 1500, el único que corresponde es el municipio de Zonte así como también se agrega el municipio menor un ejemplo como Chicontepec la cual nos dice que es falsa.

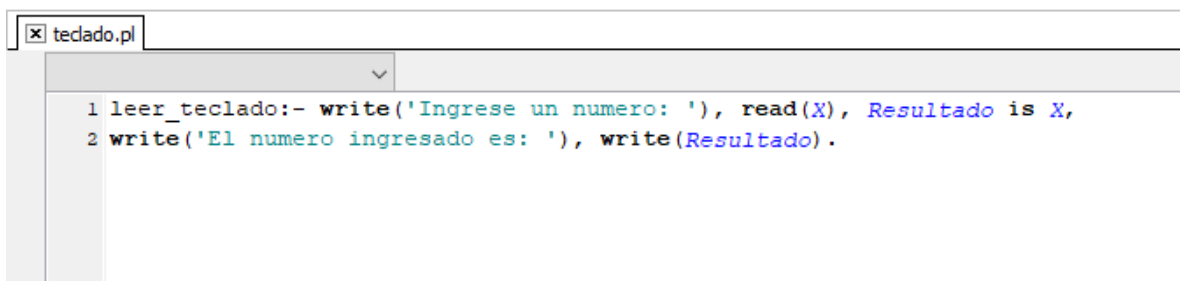
```
?-  
% d:/RCamelia/Pictures/programacion logica y funcional/municipios.pl compiled 0.00 sec, 0 clauses  
?- municipio_menor(chicon).  
false.  
  
?- municipio_menor(zonte).  
true.  
  
?- municipio_menor(X).  
X = benitojuarez ;  
X = zonte.  
  
?- ;
```

El primer resultado muestra el menor que 4000 son todos los municipios.

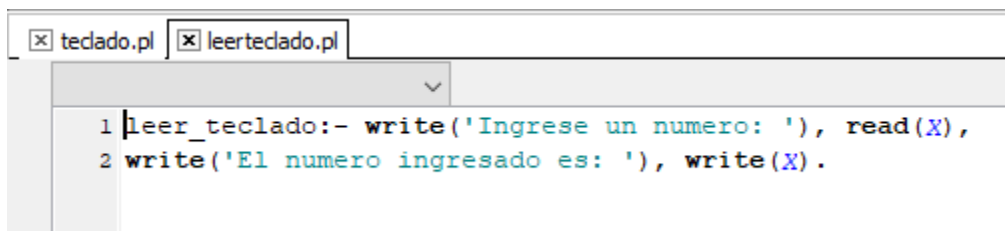
```
?-  
% d:/RCamelia/Pictures/programacion logica y funcional/municipios.pl compiled 0.00 sec, 0 clauses  
?- municipio_menor(X).  
X = benitojuarez ;  
  
?-  
% d:/RCamelia/Pictures/programacion logica y funcional/municipios.pl compiled 0.00 sec, 0 clauses  
?- municipio_menor(X).  
X = chicon ;  
X = benitojuarez ;  
X = ixhuatlan ;  
X = zonte.
```


DATOS DEL TECLADO

Las siguientes imágenes son 2 diferentes programaciones, pero con el mismo objetivo, que es pedir datos desde el teclado. Se declaran leer_teclado: write la cual nos pide ingresar un numero se refiere a que se van a ingresar datos del teclado, seguido de un mensaje, el read los lee, muestra un resultado, se vuelve a escribir write para que aparezca el mensaje de resultado y devuelve el resultado del número ingresado.



```
1 leer_teclado:- write('Ingrese un numero: '), read(X), Resultado is X,  
2 write('El numero ingresado es: '), write(Resultado).
```



```
1 leer_teclado:- write('Ingrese un numero: '), read(X),  
2 write('El numero ingresado es: '), write(X).
```

En la siguiente imagen nos muestra los resultados para poder leer el teclado de un número. Para ejecutar las instrucciones establecidas en .pl se abre el archivo en file y consulta; en este caso es: leer teclado posteriormente se muestra el mensaje para ingresar un número, se agrega cualquier número, este devuelve un mensaje indicando el número que se ingresó, finalmente se muestra un true, que significa verdad.

```
% d:/RCamelia/Pictures/programacion logica y funcional/teclado.pl compiled 0.00 sec, 0 clauses
?- leer_teclado.
Ingrese un numero: 8
|: .
El numero ingresado es: 8
true.

?- leer_teclado.
Ingrese un numero: 8.
El numero ingresado es: 8
true.

?- leer_teclado.
Ingrese un numero: 20.
El numero ingresado es: 20
true.
```

En la siguiente imagen nos muestra el error al ingresar un numero Solo está programado que funcione con números, si se agrega una letra mostrara un mensaje que indica que es error así como el siguiente ejercicio si ingresamos una letra nos marca un error la cual solo funciona con números .

```
?- leer_teclado.
Ingrese un numero: m
|: .
ERROR: Arithmetic: 'm/0' is not a function
ERROR: In:
ERROR: [9] _4796 is m
ERROR: [8] leer_teclado at d:/rcamelia/pictures/programacion logica y funcional/teclado.pl:1
ERROR: [7] <user>
?- ■
```

ÁRBOL GENEALÓGICO

En el siguiente programa agregamos los nombres de los integrantes de la familia del alumno Se declara los predicados y los argumentos, es decir, familia, seguida de los nombres de los integrantes. Para que se muestren los nombres de los integrantes se debe anotar el predicado entre paréntesis la variable X, seguida de ; punto y coma.

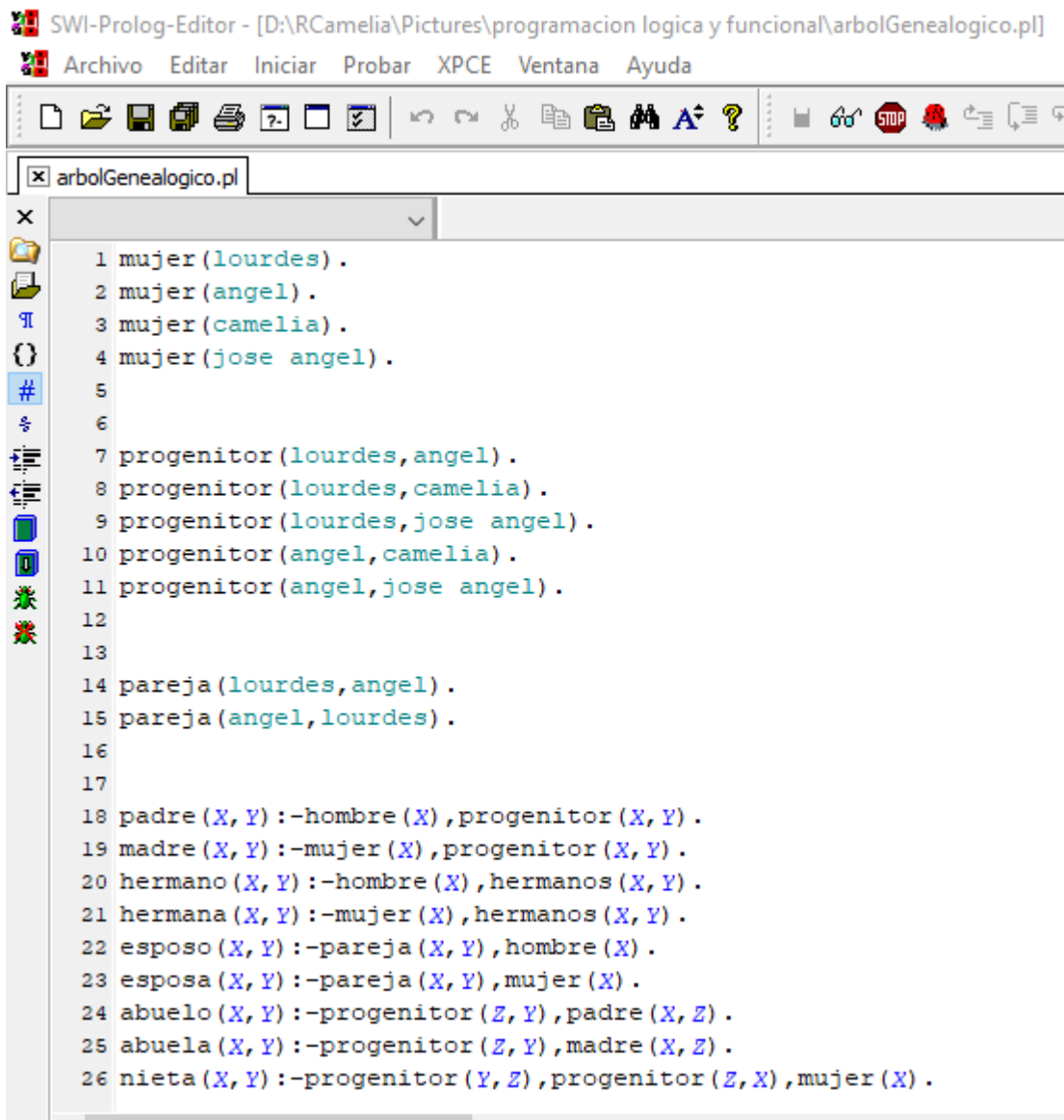
```

x arbolGenealogico.pl x ArbolGenealogico2.pl
1 familia(madre:lourdes) .
2 familia(padre:angel) .
3 familia(hermano:joseangel) .
4 familia(yo:camelia) .
5
6
7

:
% d:/RCamelia/Pictures/programacion logica y funcional/arbolGenealogico.pl
?- familia(X).
X = madre:lourdes ;
X = padre:angel ;
X = hermano:joseangel ;
X = yo:camelia.

?-
```

En la siguiente imagen nos muestra que en Otra forma de declarar el árbol genealógico es: asignar primero el género de todos los integrantes mujer(nombre). hombre(nombre). Posteriormente el progenitor, es decir, los padres seguida de los nombres de los hijos, después se agregan los nombres de las parejas primero de la mujer y después del hombre, por último, se establecen las reglas: padre debe ser hombre seguido de su progenitor, así también la madre seguida de su nombre; los datos del esposo se relacionan con la pareja, el de los abuelos se relaciona con el progenitor; la nieta se relaciona con el progenitor.



```

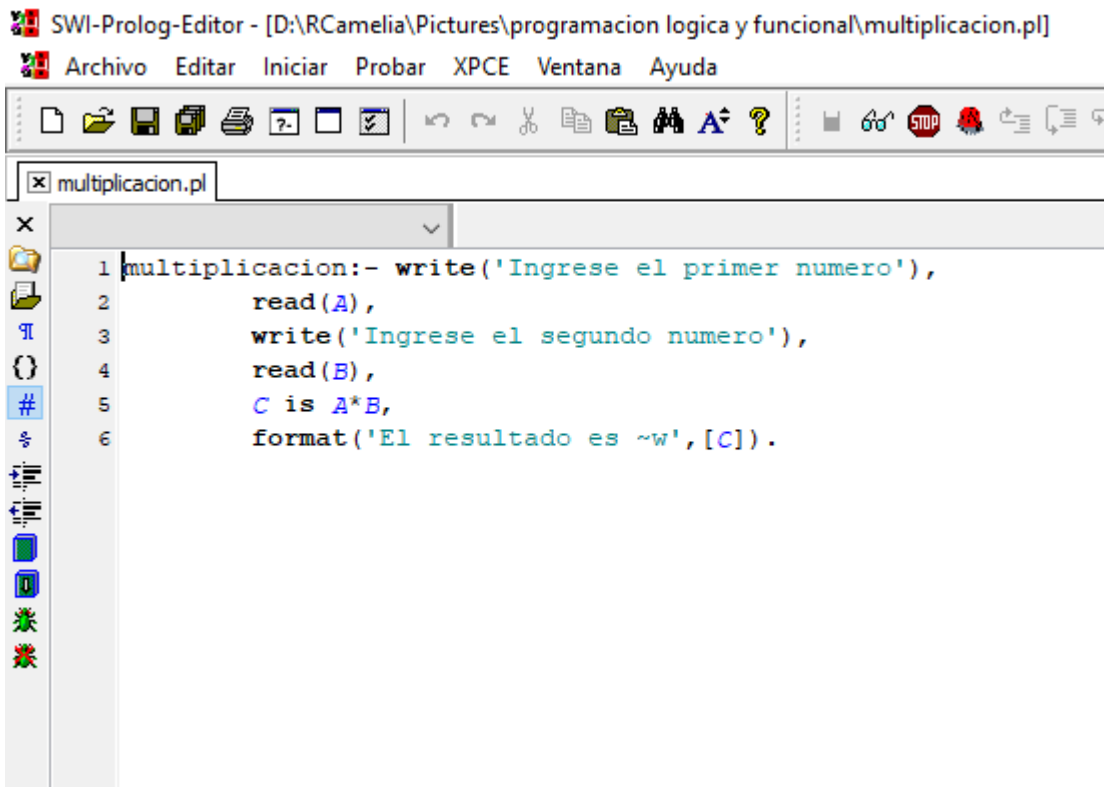
SWI-Prolog-Editor - [D:\RCamelia\Pictures\programacion logica y funcional\arbolGenealogico.pl]
Archivo  Editar  Iniciar  Probar  XPCE  Ventana  Ayuda

1 mujer(lourdes) .
2 mujer(angel) .
3 mujer(camelia) .
4 mujer(jose angel) .
5
6
7 progenitor(lourdes,angel) .
8 progenitor(lourdes,camelia) .
9 progenitor(lourdes,jose angel) .
10 progenitor(angel,camelia) .
11 progenitor(angel,jose angel) .
12
13
14 pareja(lourdes,angel) .
15 pareja(angel,lourdes) .
16
17
18 padre(X,Y):-hombre(X),progenitor(X,Y) .
19 madre(X,Y):-mujer(X),progenitor(X,Y) .
20 hermano(X,Y):-hombre(X),hermanos(X,Y) .
21 hermana(X,Y):-mujer(X),hermanos(X,Y) .
22 esposo(X,Y):-pareja(X,Y),hombre(X) .
23 esposa(X,Y):-pareja(X,Y),mujer(X) .
24 abuelo(X,Y):-progenitor(Z,Y),padre(X,Z) .
25 abuela(X,Y):-progenitor(Z,Y),madre(X,Z) .
26 nieta(X,Y):-progenitor(Y,Z),progenitor(Z,X),mujer(X) .
  
```

MULTIPLICACIÓN DE 2 NÚMEROS

La multiplicación es la operación matemática que consiste en hallar el resultado de sumar un número tantas veces como indique otro. Los factores (a y b) son los números que se multiplican. Al factor a también se le llama multiplicando. Al factor b también se le llama multiplicador.

Se declara el predicado llamado multiplicación, se agrega la palabra write seguido de un mensaje que se podrá visualizar en pantalla que indica que se debe ingresar el primer número, se agrega read (A) aquí se leen los datos del número, posteriormente se realiza lo mismo para el segundo número, después se declara la C como producto de A*B (multiplicación), con el format del mensaje del resultado de C



```
SWI-Prolog-Editor - [D:\RCamelia\Pictures\programacion logica y funcional\multiplicacion.pl]
Archivo  Editar  Iniciar  Probar  XPCE  Ventana  Ayuda

multiplicacion.pl
1 multiplicacion:- write('Ingrese el primer numero'),
2   read(A),
3   write('Ingrese el segundo numero'),
4   read(B),
5   C is A*B,
6   format('El resultado es ~w',[C]).
```

La siguiente imagen muestra el resultado de algunas multiplicaciones de algunos números, primero se debe agregar en el nombre multiplicación, el primer número, el segundo y muestra el resultado con la palabra true, está diseñado para numero si se llega a agregar una letra no va a funcionar, va a arrojar un mensaje de error.

```
% d:\RCamelia\Pictures\programacion logica y funcional\multiplicacion.pl compiled 0.00 sec, 1 clauses
?- multiplicacion.
Ingrese el primer numero 4.
Ingrese el segundo numero|: 4.
El resultado es 16
true.

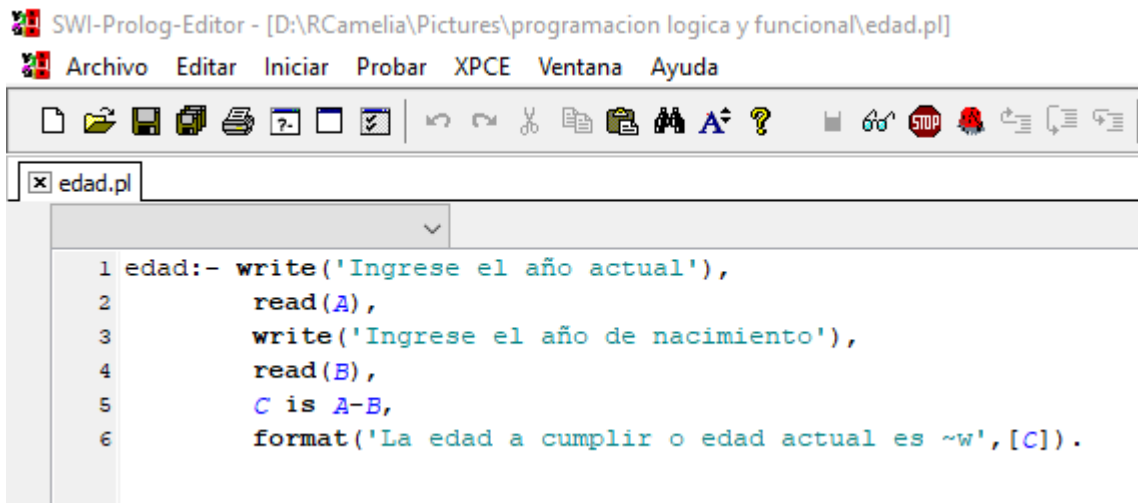
?- multiplicacion.
Ingrese el primer numero 10.
Ingrese el segundo numero|: 5.
El resultado es 50
true.

?- multiplicacion.
Ingrese el primer numero 100.
Ingrese el segundo numero|: 30.
El resultado es 3000
true.

?- multiplicacion.
Ingrese el primer numero 2.
Ingrese el segundo numero|: q
|: .
ERROR: Arithmetic: 'q/0' is not a function
ERROR: In: [9] _10296 is 2*q
ERROR: [8] multiplicacion at d:\rcamelia/pictures/programacion logica y funcional/multiplicacion.pl:5
ERROR: [7] <user>
?-
```

EDAD ACTUAL

En el siguiente programa nos muestra que debemos de agregar sobre calcular la edad actual Se agrega el predicado de edad, también el write con un mensaje de agregar el año actual, se lee el mensaje, después sigue el año de nacimiento, la operación realizar es que A-B el resultado arrojará la edad de la persona.



```
SWI-Prolog-Editor - [D:\RCamelia\Pictures\programacion logica y funcional\edad.pl]
Archivo Editar Iniciar Probar XPCE Ventana Ayuda

1 edad:- write('Ingrese el año actual'),
2       read(A),
3       write('Ingrese el año de nacimiento'),
4       read(B),
5       C is A-B,
6       format('La edad a cumplir o edad actual es ~w',[C]).
```

La siguiente imagen muestra algunos ejemplos de las edades de algunas personas la cual nos muestra que ingresemos el año actual, ingresar el año de nacimiento y al finalizar nos muestra la edad a cumplir la edad que es 22 de igual manera podemos calcular la edad de las personas solo ingresando los años y nos muestra el resultado. Si se agregamos letras mandara un error porque está diseñado para números.

 SWI-Prolog (AMD64, Multi-threaded, version 8.0.3)

File Edit Settings Run Debug Help

?-

% d:/RCamelia/Pictures/programacion logica y funcional/edad.pl compiled 0.00 sec, 1 clauses

?- edad.

Ingrese el año actual 2020.

Ingrese el año de nacimiento|: 1998.

La edad a cumplir o edad actual es 22

true.

?- edad.

Ingrese el año actual 2020.

Ingrese el año de nacimiento|: 2002.

La edad a cumplir o edad actual es 18

true.

?- edad.

Ingrese el año actual 2020.

Ingrese el año de nacimiento|: 1977.

La edad a cumplir o edad actual es 43

true.

?- edad.

Ingrese el año actual 2020.

Ingrese el año de nacimiento|: 1974.

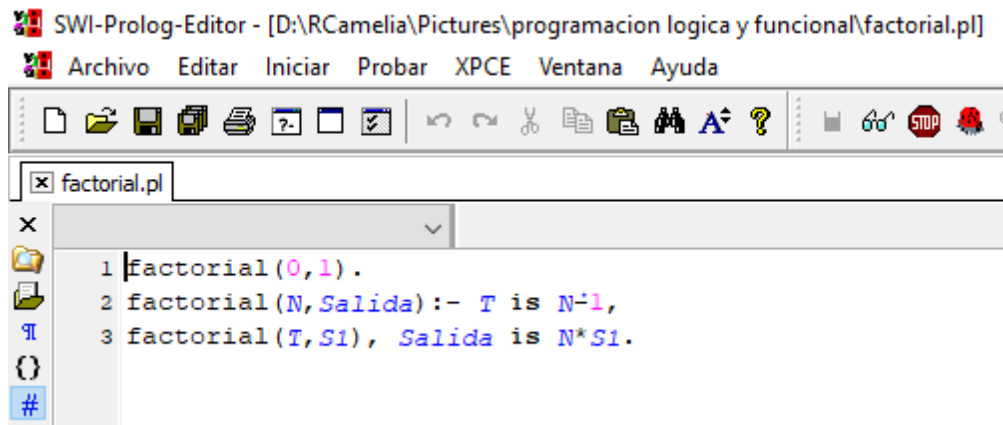
La edad a cumplir o edad actual es 46

true.

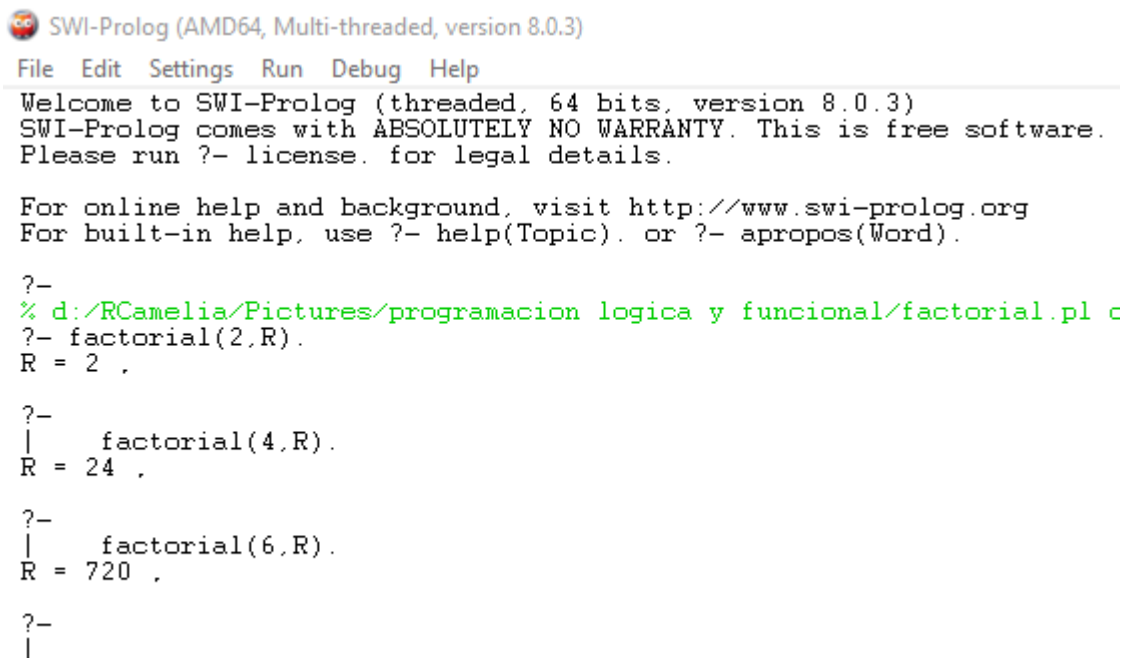
?-

FACTORIAL DE UN NUMERO

1. Se tienen dos listas con el mismo número de objetos, escriba un programa en prolog que tome alternativamente elementos de ambas listas para construir una nueva lista con los elementos de ambas.
2. Desarrolle un programa que invierta el orden de los elementos de una lista dada en factorial.



En la siguiente imagen nos muestra el resultado de un numero factorial

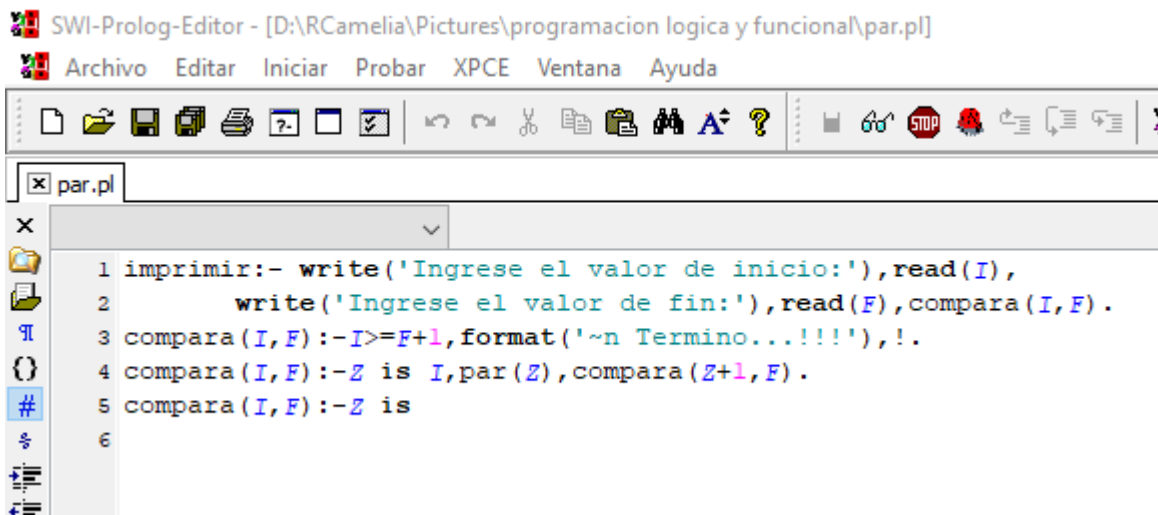



Al ingresar una letra en el programa factorial nos muestra el error.

```
% d:/RCamelia/Pictures/programacion logica y funcional/factorial.pl com
| factorial(R,R).
ERROR: Arguments are not sufficiently instantiated
ERROR: In:
ERROR: [8] factorial(_4416,_4418)
ERROR: [7] <user>
```

NUMERO PAR

Número Par. Cualquier entero que puede ser dividido exactamente por 2. El último dígito será 0, 2, 4, 6 u 8. Ejemplo: -24, 0, 6 y 38 son todos números pares. Primero se piden los numero de el rango (iniciar, finalizar), se leen con la letra de la variable F e I, estas se comparan (I, F), se compara para finalizar el bucle, la segunda comparación es para que se imprima el numero par y se aumente Z en 1, la tercera comparación para que cuando sea par se siga aumentando, por último, se comprueba los números pares a mostrar,



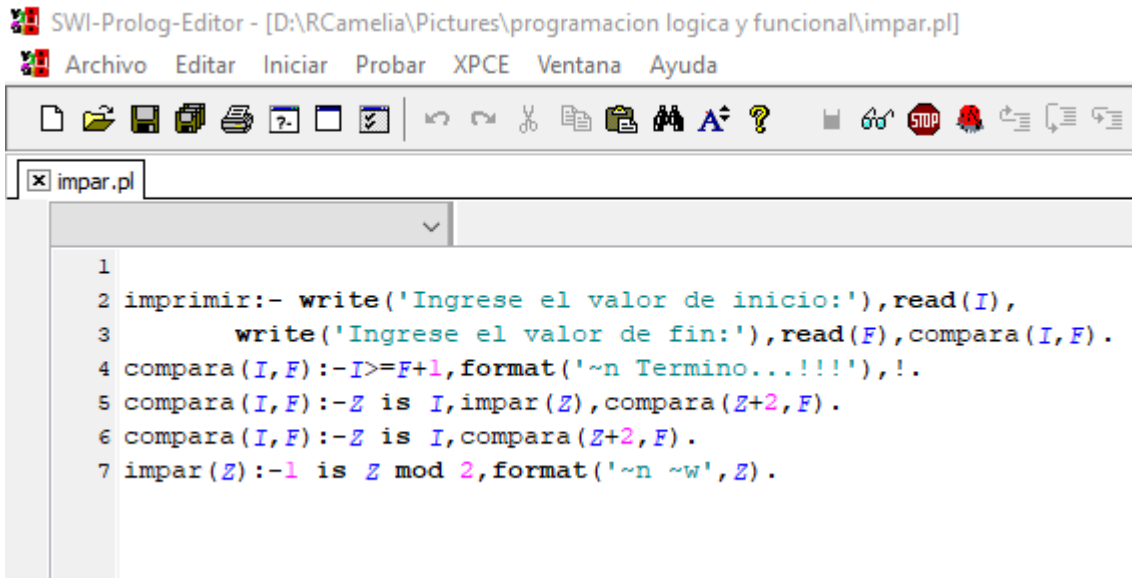
 SWI-Prolog (AMD64, Multi-threaded, version 8.0.3)

File Edit Settings Run Debug Help

```
?-  
% d:/RCamelia/Pictures/programacion logica y funcional/par.pl  
?- imprimir.  
Ingrese el valor de inicio:20.  
Ingrese el valor de fin:|: 100.  
  
20  
22  
24  
26  
28  
30  
32  
34  
36  
38  
40  
42  
44  
46  
48  
50  
52  
54  
56  
58  
60  
62  
64  
66  
68  
70  
72  
74  
76  
78  
80  
82  
84  
86  
88  
90  
92  
94  
96  
98  
100  
Termino...!!!  
true .
```

NÚMERO IMPAR

Número Impar. Cualquier entero que no puede ser dividido exactamente por 2. El último dígito será 1, 3, 5, 7 o 9. Ejemplo: -3, 1, 7 y 35 son todos números impares al imprimir con write ingresamos el valor de inicio y read para que pueda leer el valor de final del numero impar y al ingresar nos muestra el resultado lo cual se compara.



```
1
2 imprimir:- write('Ingrese el valor de inicio:'),read(I),
3           write('Ingrese el valor de fin:'),read(F),compara(I,F).
4 compara(I,F):-I>=F+1,format('~n Termino...!!!'),!.
5 compara(I,F):-Z is I,impar(Z),compara(Z+2,F).
6 compara(I,F):-Z is I,compara(Z+2,F).
7 impar(Z):-1 is Z mod 2,format('~n ~w',Z).
```

Los números impares son aquellos números enteros que no son números pares y por tanto no son múltiplos de 2. Los primeros números positivos impares son: 1, 3, 5, 7, 9... Sumando o restando 2 a un número impar se obtiene otro número impar. Sumando o restando una unidad a un número impar se obtiene un número par.

```
?-  
% d:/RCamelia/Pictures/programacion logica y funcional/impar.pl  
?- imprimir.  
Ingrese el valor de inicio:1.  
Ingrese el valor de fin:|: 50.  
  
1  
3  
5  
7  
9  
11  
13  
15  
17  
19  
21  
23  
25  
27  
29  
31  
33  
35  
37  
39  
41  
43  
45  
47  
49  
Termino...!!!  
true .  
?-
```

CONCLUSIÓN

En prolog tenemos dos piezas elementales para construir nuestros programas. Los hechos y los predicados. Los hechos son axiomas o constantes que el sistema tomará como ciertos y usará, junto con los predicados, para encontrar soluciones a las peticiones que le planteemos.

Está basado en dos ventanas: o La ventana principal, con una línea donde se ejecutan los objetivos o La ventana de edición, donde se editan y compilan los programas o La ventana principal siempre existe, la de edición sólo cuando se está usando.

Los programas Prolog hay que editarlos con un editor de texto. Se puede usar el Notepad, pero no conviene, es bastante limitado.

Hay uno que se llama emacs que viene con el SWI. También se pueden usar editores de texto de uso general, p.ej. el UltraEdit. En la página de editores contamos algunos tipos sobre cómo trabajar con los editores. Conviene guardar los archivos con extensión .pl.