

INSTITUTO TECNOLÓGICO SUPERIOR DE CHICONTEPEC.

INGENIERIA EN SISTEMAS COMPUTACIONALES.

ASIGNATURA:

Programación Lógica y Funcional.

ALUMNO:

Flor Hernández Cruz.

DOCENTE:

Ingeniero Efrén Flores Cruz.

UNIDAD 3:

Programación Lógica.

TRABAJO:

Ejercicios SWI.

SEMESTRE:

8º

FECHA DE ENTREGA:

11 de mayo 2020.

Contenido

INTRODUCCION	3
DESARROLLO	4
FUNCIONAMIENTO CON LOS ARCHIVOS .PL	4
EJERCICIO DE ALUMNOS ISC8	6
EJERCICIO DE MUNICIPIOS	8
EJERCICIO DE DATOS DEL TECLADO	10
TAREAS	12
ÁRBOL GENEALÓGICO DEL ALUMNO.	12
MULTIPLICACIÓN DE 2 NÚMEROS	15
CALCULAR LA EDAD ACTUAL	16
FACTORIAL DE UN NUMERO	17
MUESTRA NÚMERO IMPAR	20
CONCLUSION	22

INTRODUCCION

El presente trabajo representa la unidad 3 de programación lógica y funcional, con el tema central de programación lógica, se desglosan los siguientes subtemas: repaso de la lógica de primer orden, unificación y resolución, cláusulas de Horn, resolución SLD, programación lógica de cláusulas Horn.

Para este trabajo se usó un software de aplicación llamado SWI-Prolog, es una implementación en código abierto del lenguaje de programación Prolog. Su autor principal es Jan Wielemaker. En desarrollo ininterrumpido desde 1987.

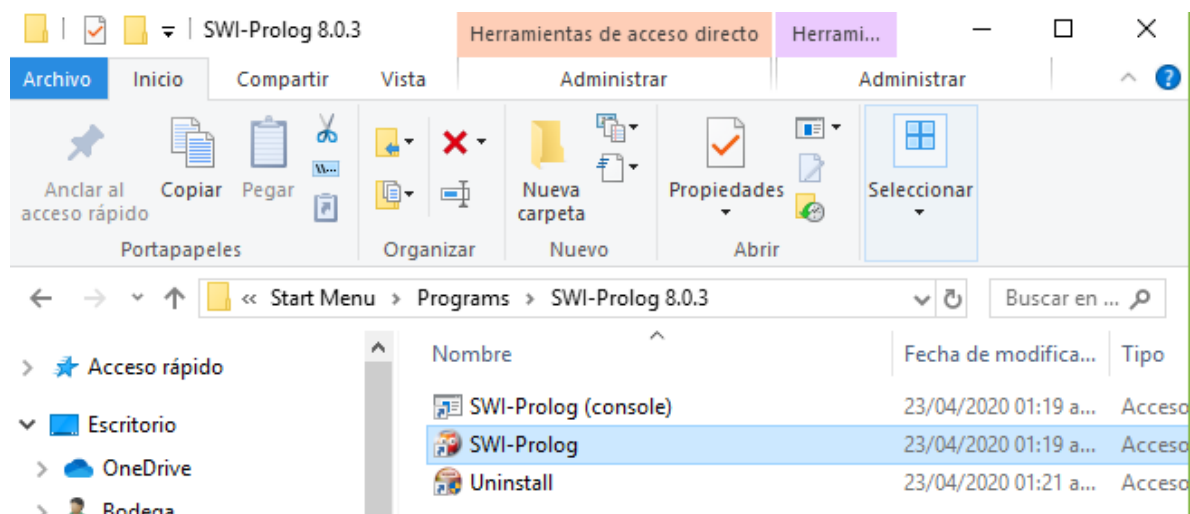
Esta actividad consiste en la resolución de algunos ejercicios usando un archivo con extensión de .PL, aquí está ubicado la programación de que se desea resolver. Posteriormente este archivo se abre en SWI-Prolog.

Es muy importante tomar en cuenta las reglas específicas de este lenguaje, porque si no se respeta adecuadamente no se podrá ejecutar correctamente, mandara un mensaje de error.

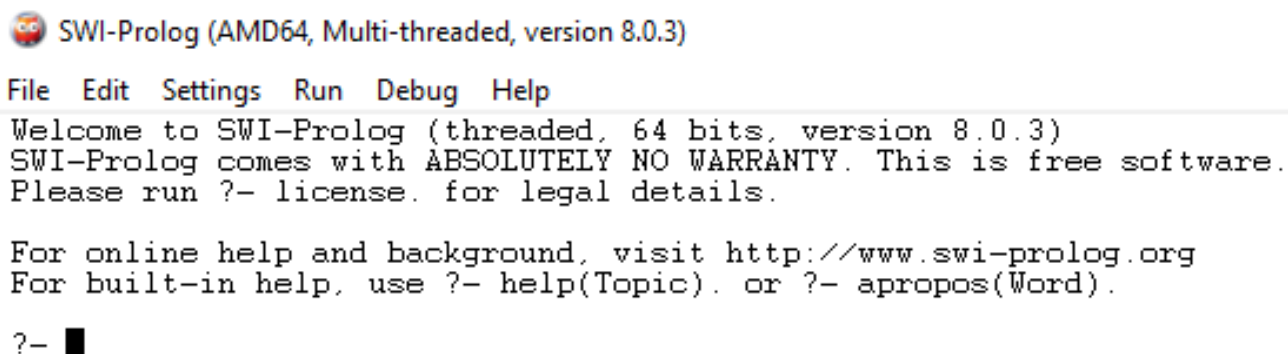
DESARROLLO

FUNCIONAMIENTO CON LOS ARCHIVOS .PL

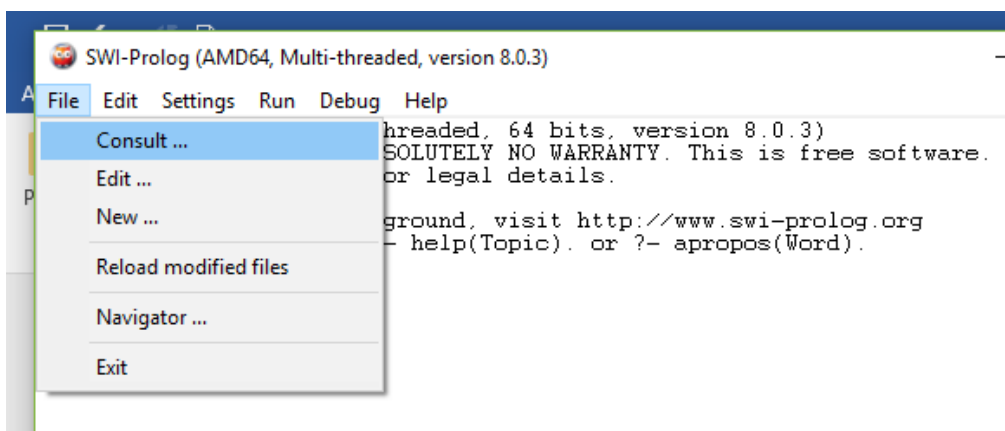
Primero se abre el programa de SWI-Prolog



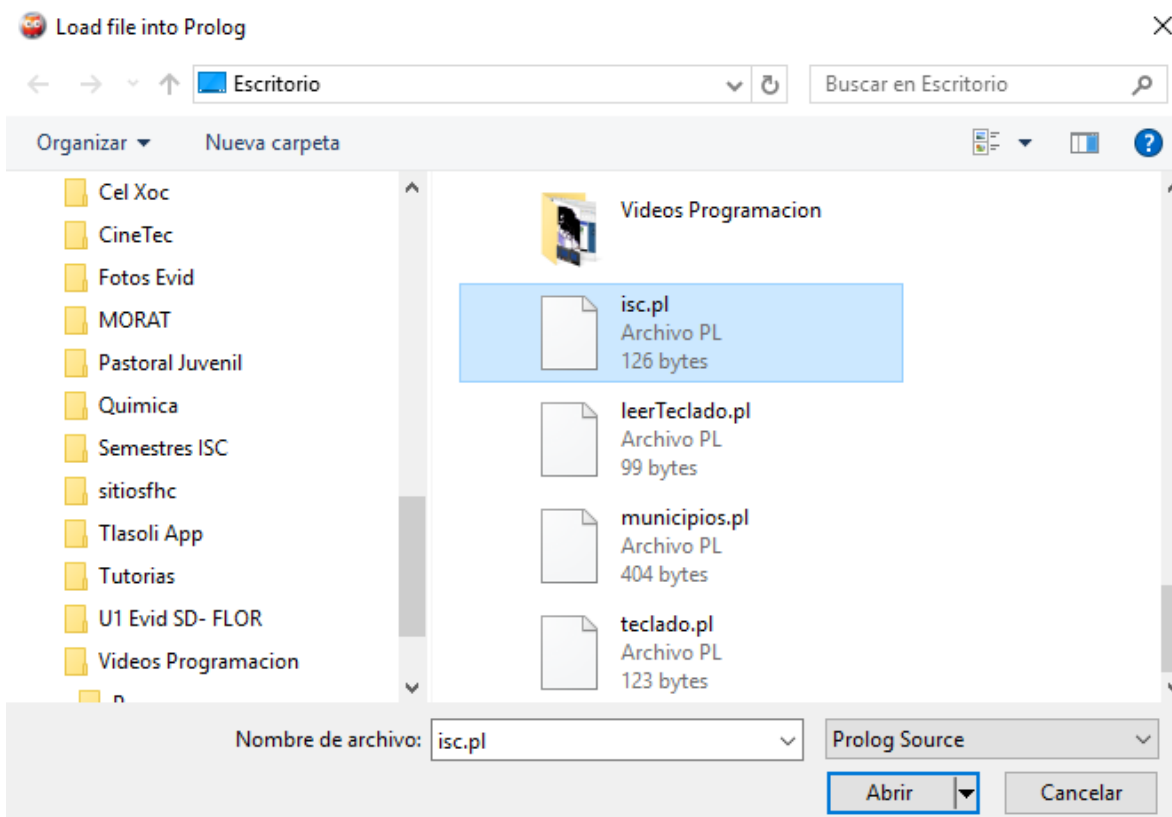
En la siguiente imagen se muestra el área de trabajo de SWI-Prolog.



Para la realización de los siguientes ejercicios, las instrucciones (programación) se va a realizar en un bloc de notas, posteriormente este 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.

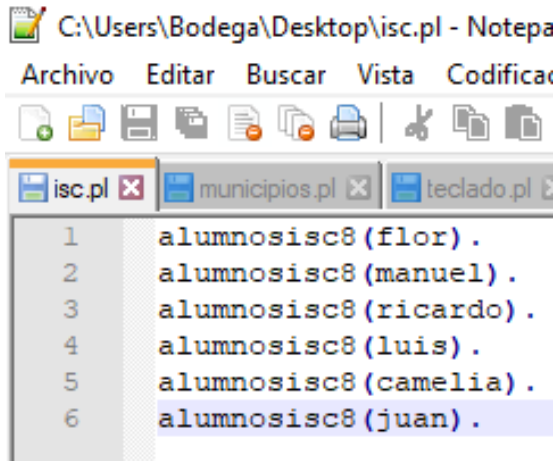


Se selecciona el archivo que se desea ejecutar, clic en la opción de abrir.




EJERCICIO DE ALUMNOS ISC8

Se realiza la programación en algún editor de texto, este ejercicio indica todos los alumnos de la carrera de sistemas de octavo semestre.



```
C:\Users\Bodega\Desktop\isc.pl - Notepad
Archivo  Editar  Buscar  Vista  Codifica
isc.pl  municipios.pl  teclado.pl
1  alumnosisc8(flor).
2  alumnosisc8(manuel).
3  alumnosisc8(ricardo).
4  alumnosisc8(luis).
5  alumnosisc8(camelia).
6  alumnosisc8(juan).
```

Se abre el archivo de alumnosisc8, se pregunta si la alumna flor es de ese grupo con la siguiente instrucción: `alumnosisc8(flor).` 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).

 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).

?-
% c:/Users/Bodega/Desktop/isc.pl compiled 0.00 sec, 6 clauses
?- alumnosisc8(flor).
true.

?- alumnosisc8(manuel).
true.

?- alumnosisc8(pedro).
false.
```

Para mostrar los nombres de todos los integrantes de este grupo se agrega el siguiente comando: `alumnosisc8(X)`. se agrega la variable `X` por que no se sabe el número de integrantes, posteriormente se agrega el; para se sigan mostrando los demás nombres y así hasta finalizar los nombres de los alumnos.

 **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).
```

```
?-
```

```
% c:/Users/Bodega/Desktop/isc.pl compiled 0.00 sec, 6 clauses
```

```
?-
```

```
| alumnosisc8(X).
```

```
X = flor ;
```

```
X = manuel ;
```

```
X = ricardo ;
```

```
X = luis ;
```

```
X = camelia ;
```

```
X = juan.
```

```
?- ■
```

EJERCICIO DE MUNICIPIOS

USANDO REGLAS

Se agregan primero los nombres de los municipios, posteriormente se agregan los límites de los municipios respectivamente a su territorio con el nombre de los municipios, posteriormente 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.

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

```
:  
% c:/Users/Bodega/Desktop  
?- 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.

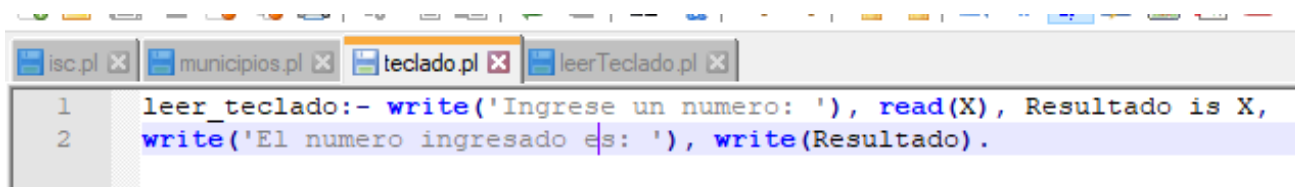
```
:  
% c:/Users/Bodega/Desktop/municipios.:  
?- municipio_menor(ixhuatlan).  
false.  
  
?- municipio_menor(zonte).  
true.  
  
?-  
|  
| municipio_menor(zonte).  
true.  
  
?- municipio_menor(X).  
X = zonte.
```

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

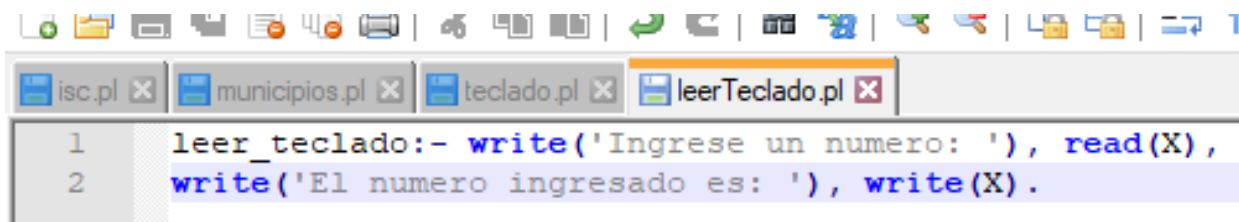
```
!-  
% c:/Users/Bodega/Desktop/mun  
?- municipio_menor(X).  
X = benitojuarez .  
  
?-  
% c:/Users/Bodega/Desktop/mun  
?- municipio_menor(X).  
X = chicon ;  
X = benitojuarez ;  
X = ixhuatlan ;  
X = zonte.
```

EJERCICIO DE DATOS DEL TECLADO

Las siguientes capturas son 2 diferentes programaciones, pero con el mismo objetivo, que es pedir datos desde el teclado. Se declaran leer _teclado: write 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 numero 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).
```

Para ejecutar las instrucciones establecidas en .pl se abre el archivo en file y consult; se debe escribir lo que se declaró en el documento, 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.

```
.  
% c:/Users/Bodega/Desktop/tecl  
?- leer_teclado.  
Ingrese un numero: 10.  
El numero ingresado es: 10  
true.  
  
?- leer_teclado.  
Ingrese un numero: 30.  
El numero ingresado es: 30  
true.  
  
?- leer_teclado.  
Ingrese un numero: 24.  
El numero ingresado es: 24  
true.
```

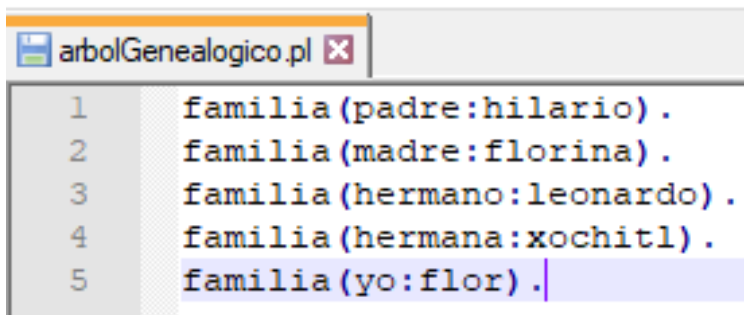
Solo está programado que funcione con números, si se agrega una letra mostrara un mensaje que indica que es error.

```
?- leer_teclado.  
Ingrese un numero: f.  
  
ERROR: Arithmetic: 'f/0' is not a function  
ERROR: In:  
ERROR: [9] _4982 is f  
ERROR: [8] leer_teclado at c:/users/bodega/desktop/teclado.pl:1  
ERROR: [7] <user>  
?- ■
```

TAREAS

ÁRBOL GENEALÓGICO DEL ALUMNO.

Se declara los predicados y los argumentos, es decir, familia, seguida de los nombres de los integrantes.



```
1 familia(padre:hilario).
2 familia(madre:florina).
3 familia(hermano:leonardo).
4 familia(hermana:xochitl).
5 familia(yo:flor).
```

Para que se muestren los nombres de los integrantes se debe anotar el predicado entre paréntesis la variable X, seguida de ; punto y coma.

```
% c:/Users/Bodega/Desktop/Archi
?- familia(X).
X = padre:hilario ;
X = madre:florina ;
X = hermano:leonardo ;
X = hermana:xochitl ;
X = yo:flor.
```

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.

```

arbolGenealogico.pl x factorial.pl x multiplicacion.pl x edad.pl x par
1  mujer(florina) .
2  mujer(berna) .
3  mujer(flor) .
4  mujer(xochitl) .
5  mujer(karla) .
6  hombre(hilario) .
7  hombre(leonardo) .
8
9  progenitor(florina,leonardo) .
10 progenitor(florina,flor) .
11 progenitor(florina,xochitl) .
12 progenitor(hilario,leonardo) .
13 progenitor(hilario,flor) .
14 progenitor(hilario,xochitl) .
15 progenitor(leonardo,karla) .
16 progenitor(berna,karla) .
17
18 pareja(florina,hilario) .
19 pareja(hilario,florina) .
20 pareja(berna,leonardo) .
21 pareja(leonardo,berna) .
22
23 padre(X,Y):-hombre(X),progenitor(X,Y) .
24 madre(X,Y):-mujer(X),progenitor(X,Y) .
25 hermano(X,Y):-hombre(X),hermanos(X,Y) .
26 hermana(X,Y):-mujer(X),hermanos(X,Y) .
27 esposo(X,Y):-pareja(X,Y),hombre(X) .
28 esposa(X,Y):-pareja(X,Y),mujer(X) .
29 abuelo(X,Y):-progenitor(Z,Y),padre(X,Z) .
30 abuela(X,Y):-progenitor(Z,Y),madre(X,Z) .
31 nieta(X,Y):-progenitor(Y,Z),progenitor(Z,X),mujer(X) .

```

Para ingresar los parentescos se agrega el predicado de abuela, por ejemplo, seguida del nombre de ella, el nombre de la nieta si es correcto enviara un mensaje de true, si los datos de algún familiar no son los correctos como el caso de que mi mama sea esposa de mi hermano mandara un mensaje de false (falso)

```
% c:/Users/Bodega/Desktop/Archivos PL -FLOR/aG.pl compiled 0.00 sec,  
6 clauses  
?- abuela(florina,karla).  
true .  
  
?- abuelo(hilario,karla).  
true .  
  
?- esposa(florina,leonardo).  
false.  
  
?- esposa(florina,hilario).  
true.  
  
?- esposo(leonardo,berna).  
true.
```

Las siguientes imágenes muestran los datos de mama Florina y papa Hilario con sus respectivos hijos: flor, Xóchitl, Leonardo, se agregó a Karla, pero esta arroja como falso debido a que ella es la nieta.

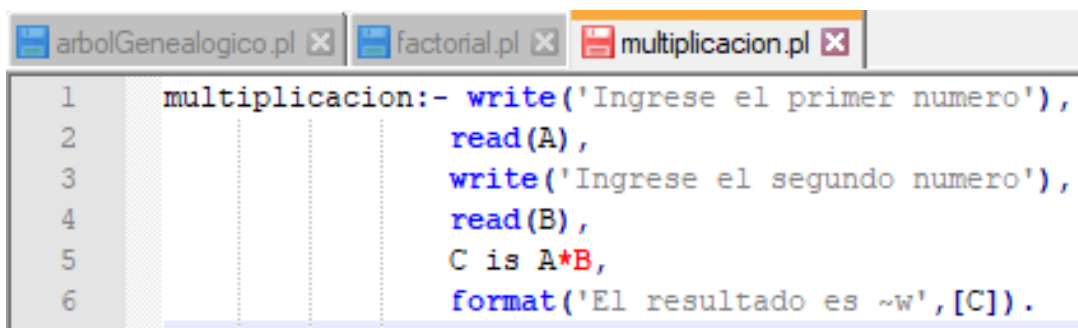
```
% c:/Users/Bodega/Desktop/Archivos PL -FLOR/aG.pl compiled 0.00 sec,  
0 clauses  
?- madre(florina,flor).  
true .  
  
?- madre(florina,xochitl).  
true .  
  
?- madre(florina,karla).  
false.  
  
?- madre(florina,leonardo).  
true .  
  
?- padre(hilario,flor).  
true.  
  
?- padre(hilario,xochitl).  
true.  
  
?- padre(hilario,leonardo).  
true.  
  
?- padre(hilario,karla).  
false.  
  
?- padre(hilario,berna).  
false.
```

Las siguientes imágenes muestran los resultados de la única nieta que existe en la familia que es Karla, de mamá Florina y papá Hilario; se agrega la instrucción de que Karla es nieta de flor, arroja falso debido a que yo (flor) soy su tía.

```
?- nieta(karla, florina).  
true .  
  
?- nieta(karla, hilario).  
true .  
  
?- nieta(karla, flor).  
false .
```

MULTIPLICACIÓN DE 2 NÚMEROS

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



```
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 el predicado, el primer número, el segundo y muestra el resultado con la palabra true, está diseñado para número si se llega a agregar una letra no va a funcionar, va a arrojar un mensaje de error.

```
% c:/Users/Bodega/Desktop/Archivos PL -FLOR/mu
?- multiplicacion.
Ingrese el primer numero8.
Ingrese el segundo numero|: 8.
El resultado es 64
true.

?- multiplicacion.
Ingrese el primer numero4.
Ingrese el segundo numero|: 9.
El resultado es 36
true.

?- multiplicacion.
Ingrese el primer numero12.
Ingrese el segundo numero|: 4.
El resultado es 48
true.

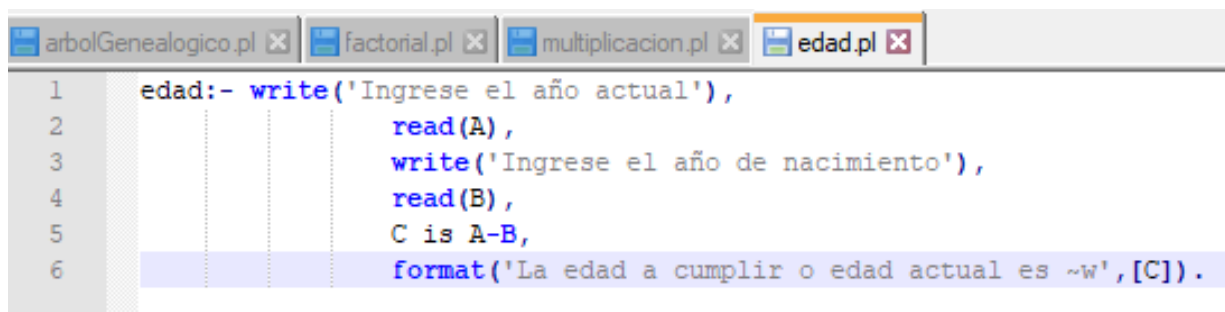
?- multiplicacion.
Ingrese el primer numero11.
Ingrese el segundo numero|: 39.
El resultado es 429
true.

?- multiplicacion.
Ingrese el primer numero4.
Ingrese el segundo numero|: n.

ERROR: Arithmetic: 'n/0' is not a function
ERROR: In:
ERROR:      [9] _1996 is 4*n
ERROR:      [8] multiplicacion at c:/users/bodeg
ERROR:      [7] <user>
```

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.



```
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.

Si se agrega letras mandara un error por que está diseñado para números.

```
% c:/Users/Bodega/Desktop/Archivos PL -FLOR/edad.pl compiled 0.00 sec, 0 clauses
?- edad.
Ingrese el año actual2020.
Ingrese el año de nacimiento|: 1997.
La edad a cumplir o edad actual es 23
true.

?- edad.
Ingrese el año actual2020.
Ingrese el año de nacimiento|: 2008.
La edad a cumplir o edad actual es 12
true.

?- edad.
Ingrese el año actual2020.
Ingrese el año de nacimiento|: 1967.
La edad a cumplir o edad actual es 53
true.

?- edad.
Ingrese el año actual2020.
Ingrese el año de nacimiento|: 1983.
La edad a cumplir o edad actual es 37
true.

?- edad.
Ingrese el año actualfff.
Ingrese el año de nacimiento|: tr.

ERROR: Arithmetic: `tr/0' is not a function
ERROR: In:
ERROR:      [9] _3880 is fff-tr
ERROR:      [8] edad at c:/users/bodega/desktop/archivos pl -flor/edad.pl:5
```

FACTORIAL DE UN NUMERO

Factorial de 0 es 1, posteriormente se llama recursivamente a factorial, N es el número que se va a solicitar para calcular la factorial, la salida es el que se va a multiplicar, T es igual a N-1, por que la factorial se calcula multiplicando todos los números de forma descendente, por ejemplo, factorial de 5 se calcularía como $5*4*3*2*1$ al número a calcular se resta un número así sucesivamente hasta llegar al 1. Después la factorial se agrega T que es convertido en N-1, la salida es igual a $N*S1$

```
arbolGenealogico.pl x factorial.pl x
1 factorial(0,1).
2 factorial(N,Salida):- T is N-1,
3 factorial(T,S1), Salida is N*S1.
```

La siguiente imagen muestra la factorial de algunos números, se agrega primero el predicado entre paréntesis el número que se desea calcular seguido de coma y la letra R que indica el resultado finalizando con el punto.

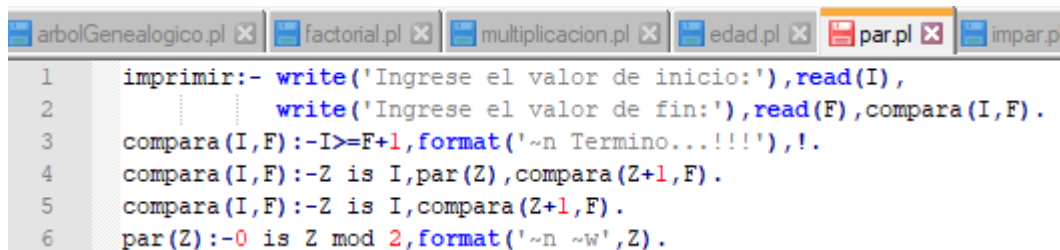
```
.  
% c:/Users/Bodega/Desktop/  
?- factorial(5,R).  
R = 120 .  
  
?- factorial(2,R).  
R = 2 .  
  
?- factorial(4,R).  
R = 24 .  
  
?- factorial(8,R).  
R = 40320 .
```

Si se agrega una letra no se podrá procesar por que no admite este tipo de caracteres.

```
?- factorial(e,R).  
factorial(e,R).  
ERROR: Stack limit (1.0Gb) exceeded  
ERROR: Stack sizes: local: 0.8Gb, global: 0.2Gb, trail: 0Kb  
ERROR: Stack depth: 8,657,485, last-call: 0%, Choice points: 3  
ERROR: Possible non-terminating recursion:  
ERROR: [8,657,485] user:factorial(-8657474.281718172, _51944948)  
ERROR: [8,657,484] user:factorial(-8657473.281718172, _51944968)  
?- ERROR: Stack limit (1.0Gb) exceeded  
ERROR: Stack sizes: local: 0.8Gb, global: 0.2Gb, trail: 0Kb  
ERROR: Stack depth: 8,657,485, last-call: 0%, Choice points: 3  
ERROR: Possible non-terminating recursion:  
ERROR: [8,657,485] user:factorial(-8657474.281718172, _51944948)  
ERROR: [8,657,484] user:factorial(-8657473.281718172, _51944968)  
?- ■
```

MUESTRA UN NUMERO PAR

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,



```
1  imprimir:- write('Ingrese el valor de inicio:'),read(I),
2             write('Ingrese el valor de fin:'),read(F),compara(I,F).
3  compara(I,F):-I>=F+1,format('~n Terminó...!!!'),!.
4  compara(I,F):-Z is I,par(Z),compara(Z+1,F).
5  compara(I,F):-Z is I,compara(Z+1,F).
6  par(Z):-0 is Z mod 2,format('~n ~w',Z).
```

En la siguiente imagen se muestra un rango de números pares, se debe escribir el predicado para mandar a llamar la programación establecida, manda el primer mensaje que indica el número de inicio, posteriormente el número final del rango, se muestran todos o números pares de dicho rango finalizando con un mensaje de Terminó y true.

```
% c:/Users/Bodega/Desktop/Archivos PL -
1 clauses
Can't ignore goal at this port
?- imprimir.
Ingrese el valor de inicio:10.
Ingrese el valor de fin:40.

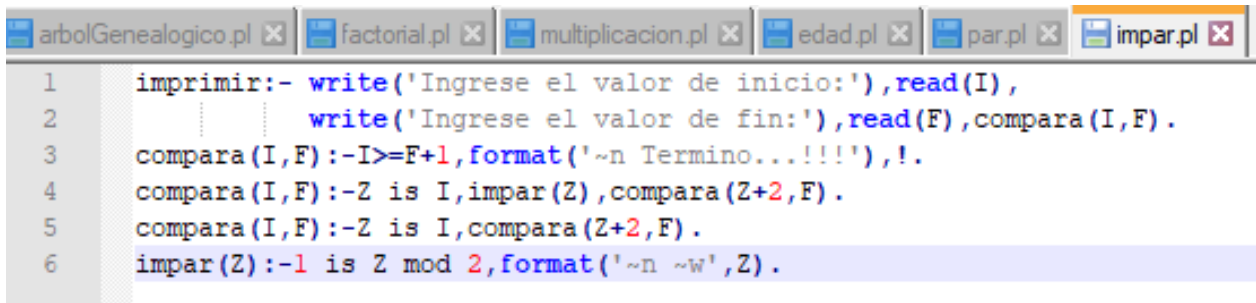
10
12
14
16
18
20
22
24
26
28
30
32
34
36
38
40
Termino...!!!
true .
```

```
?- imprimir.
Ingrese el valor de inicio:1.
Ingrese el valor de fin:|: 10.

2
4
6
8
10
Termino...!!!
true .
```

MUESTRA NÚMERO IMPAR

La programación de los numero impares es parecido al anterior de impar, los únicos datos que cambian es al momento de sumar Z en lugar de ser 1 se agrega el número 2 y se inicializa en 1.



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

Las siguientes imágenes muestran el rango de numero impares de 1-10 y 1-50

```
% c:/Users/Bodega/Desktop/Archivos 1
c. 0 clauses
Can't ignore goal at this port
?- imprimir.
Ingrese el valor de inicio:1.
Ingrese el valor de fin:|: 10.

1
3
5
7
9
Termino...!!!
true .
```

```
?- 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 .
```

CONCLUSION

SWI-Prolog es una herramienta primordial para la programación de Prolog en el ámbito funcional, los ejercicios que venían explicados en el PDF fueron fáciles de realizar, sentí un poco complicado al desarrollar los demás ejercicios, la estructura que maneja este lenguaje es muy diferente, sin embargo, si no se respeta alguna regla entonces no se podrá ejecutar adecuadamente.

Aprendí que en este software Prolog están organizados en cuatro secciones principales:

- Dominio: donde se declaran los argumentos que utilizarán los predicados.
- Predicados: donde se declaran todos los predicados no predefinidos que se utilizarán en las siguientes secciones.
- Objetivos: esta sección permite ejecutar los programas de forma no interactiva, y, por tanto, buscará la solución deseada tan pronto como se ejecute el programa. Como también es habitual usar Prolog de forma interactiva es frecuente ejecutar un programa y luego esperar a que se nos pregunte por los objetivos.
- Clausulas: donde se escriben los hechos y las reglas que conocemos del dominio.

Son indispensables para la resolución de los ejercicios.