

PROLOG

Alumno: Antonio Ramos Gonzalez

Matricula: 372576

Maestro: Carlos Gallegos

Materia: Paradigmas de la programacion

Fecha de entrega: 30 de mayo de 2024

Introduccion: PARADIGMA LOGICO

La programación lógica es un paradigma de programación en el que se utiliza la lógica formal para expresar los programas. Este enfoque se basa en la utilización de reglas y hechos lógicos para resolver problemas, en lugar de procedimientos imperativos. Para esta actividad se nos encomendo realizar codigos en el lenguaje Prolog, siguiendo las 17 secciones de su tutorial, esto con el fin de comprender la programacion logica.

Desarrollo:

La presentacion de codigo comienza apartir de la seccion 4

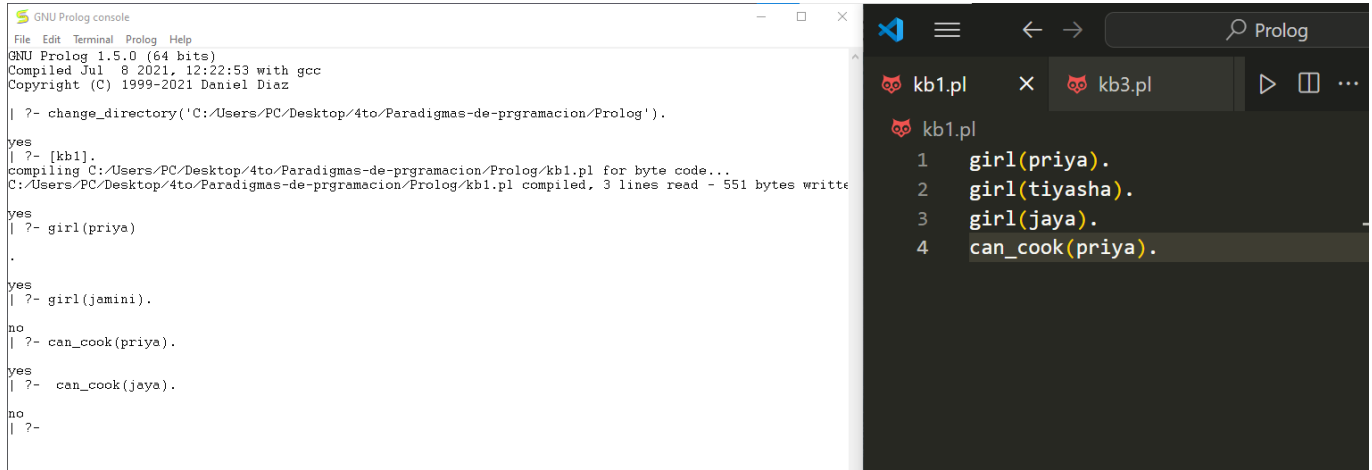
Seccion 4.- HelloWorld: nos presenta el funcionamiento basico de Prolog

```
Prolog > 🐱 HelloWorld.pl
1  #Antonio Ramos Gonzalez
2  #Seccion 4
3  main :- write('This is a sample Prolog program'),
4          write(' This program is written into hello_world.pl file').
5
6
7
8
9
10
11
12
13
```



```
GNU Prolog console
File Edit Terminal Prolog Help
GNU Prolog 1.5.0 (64 bits)
Compiled Jul  8 2021, 12:22:53 with gcc
Copyright (C) 1999-2021 Daniel Diaz
| ?- change_directory('C:/Users/PC/Desktop/4to/Paradigmas-de-prgramacion/Prolog').
yes
| ?- [helloworld].
compiling C:/Users/PC/Desktop/4to/Paradigmas-de-prgramacion/Prolog/helloworld.pl for byte code...
C:/Users/PC/Desktop/4to/Paradigmas-de-prgramacion/Prolog/helloworld.pl compiled, 2 lines read - 580 bytes written,
yes
| ?- main.
This is a sample Prolog program This program is written into hello_world.pl file
yes
| ?- |
```

Seccion 5.- Basic: Presenta hechos, sintaxis y normas del lenguaje



The image shows two windows. The left window is the GNU Prolog console, and the right window is the Prolog IDE.

GNU Prolog console output:

```
GNU Prolog 1.5.0 (64 bits)
Compiled Jul  8 2021, 12:22:53 with gcc
Copyright (C) 1999-2021 Daniel Diaz

| ?- change_directory('C:/Users/PC/Desktop/4to/Paradigmas-de-prgramacion/Prolog').
yes
| ?- [kb1].
compiling C:/Users/PC/Desktop/4to/Paradigmas-de-prgramacion/Prolog/kb1.pl for byte code...
C:/Users/PC/Desktop/4to/Paradigmas-de-prgramacion/Prolog/kb1.pl compiled, 3 lines read - 551 bytes written

yes
| ?- girl(priya).
.

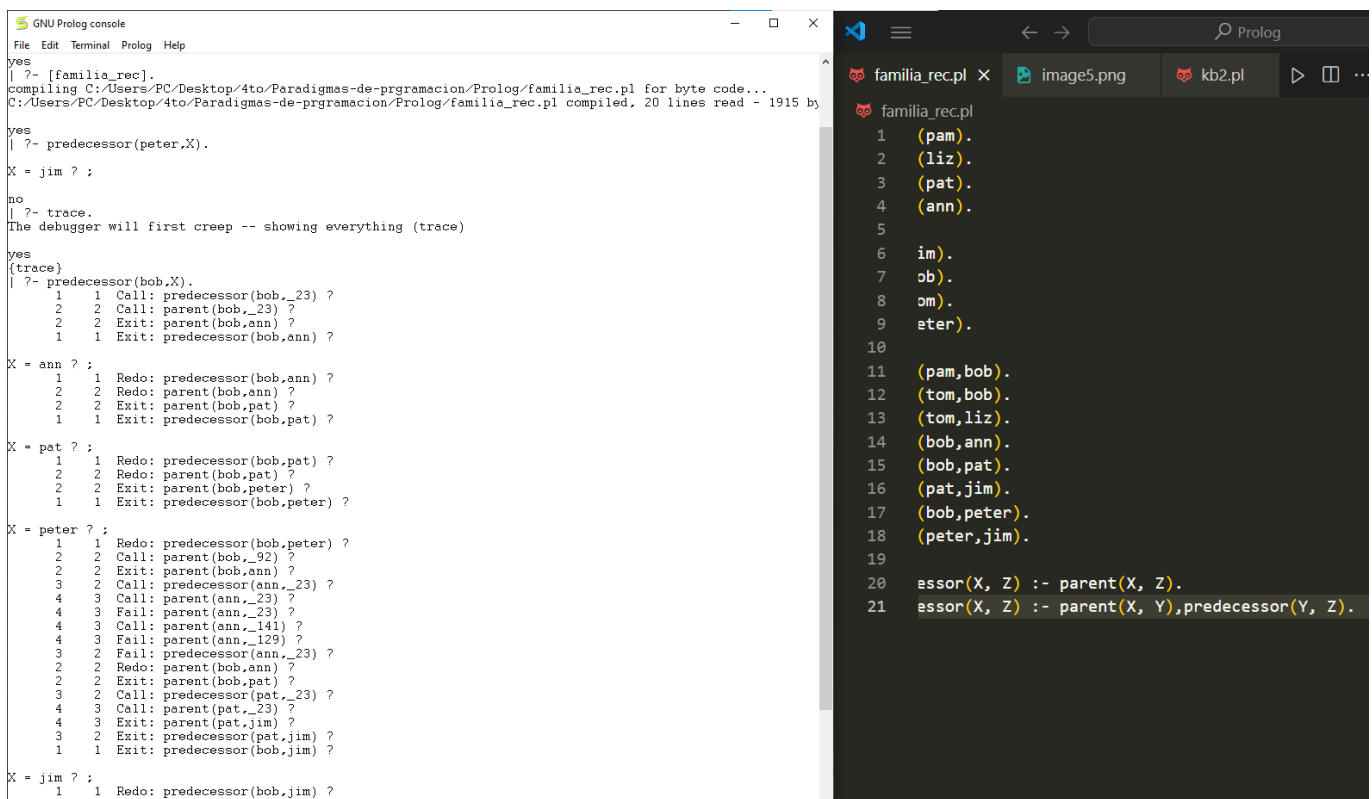
yes
| ?- girl(jemini).
no
| ?- can_cook(priya).

yes
| ?- can_cook(jaya).
no
| ?-
```

Prolog IDE output (kb1.pl):

```
1 girl(priya).
2 girl(tiyasha).
3 girl(jaya).
4 can_cook(priya).
```

Seccion 6.- Relations: nos presenta las relaciones



The image shows two windows. The left window is the GNU Prolog console, and the right window is the Prolog IDE.

GNU Prolog console output:

```
yes
| ?- [familia_rec].
compiling C:/Users/PC/Desktop/4to/Paradigmas-de-prgramacion/Prolog/familia_rec.pl for byte code...
C:/Users/PC/Desktop/4to/Paradigmas-de-prgramacion/Prolog/familia_rec.pl compiled, 20 lines read - 1915 bytes written

yes
| ?- predecessor(peter,X).
X = jim ? ;
no
| ?- trace.
The debugger will first creep -- showing everything (trace)

yes
{trace}
| ?- predecessor(bob,X).
1 1 Call: predecessor(bob,_23) ?
2 2 Call: parent(bob,_23) ?
2 2 Exit: parent(bob,ann) ?
1 1 Exit: predecessor(bob,ann) ?

X = ann ? ;
1 1 Redo: predecessor(bob,ann) ?
2 2 Redo: parent(bob,ann) ?
2 2 Exit: parent(bob,pat) ?
1 1 Exit: predecessor(bob,pat) ?

X = pat ? ;
1 1 Redo: predecessor(bob,pat) ?
2 2 Redo: parent(bob,pat) ?
2 2 Exit: parent(bob,peter) ?
1 1 Exit: predecessor(bob,peter) ?

X = peter ? ;
1 1 Redo: predecessor(bob,peter) ?
2 2 Call: parent(bob,_92) ?
2 2 Exit: parent(bob,ann) ?
3 3 Call: predecessor(ann,_23) ?
4 4 Call: parent(ann,_23) ?
4 3 Fail: parent(ann,_141) ?
4 3 Fail: parent(ann,_129) ?
3 2 Fail: predecessor(ann,_23) ?
2 2 Redo: parent(bob,ann) ?
2 2 Exit: parent(bob,pat) ?
3 3 Call: predecessor(pat,_23) ?
4 4 Call: parent(pat,_23) ?
4 3 Exit: parent(pat,jim) ?
3 2 Exit: predecessor(pat,jim) ?
1 1 Exit: predecessor(bob,jim) ?

X = jim ? ;
1 1 Redo: predecessor(bob,jim) ?
```

Prolog IDE output (familia_rec.pl):

```
1 (pam).
2 (liz).
3 (pat).
4 (ann).
5
6 im).
7 ob).
8 om).
9 eter).
10
11 (pam,bob).
12 (tom,bob).
13 (tom,liz).
14 (bob,ann).
15 (bob,pat).
16 (pat,jim).
17 (bob,peter).
18 (peter,jim).
19
20 assor(X, Z) :- parent(X, Z).
21 assor(X, Z) :- parent(X, Y),predecessor(Y, Z).
```

Seccion 7.- Data Objects: Presenta atomos y variables

The screenshot shows the GNU Prolog console on the left and a code editor on the right. The console displays the compilation of 'var_anonymous.pl' and the execution of the 'hates' predicate. The code editor shows the source code for 'var_anonymous.pl'.

```
GNU Prolog console
File Edit Terminal Prolog Help
GNU Prolog 1.5.0 (64 bits)
Compiled Jul  8 2021, 12:22:53 with gcc
Copyright (C) 1999-2021 Daniel Diaz

| ?- change_directory('C:/Users/PC/Desktop/4to/Paradigmas-de-prgramacion/Prolog').
yes
| ?- [var_anonymous].
compiling C:/Users/PC/Desktop/4to/Paradigmas-de-prgramacion/Prolog/var_anonymous.pl for byte code...
C:/Users/PC/Desktop/4to/Paradigmas-de-prgramacion/Prolog/var_anonymous.pl compiled, 3 lines read - 598 bytes
yes
| ?- hates(X,tom).
X = jim ? ;
X = peter
yes
| ?- hates(_,tom).
true ? ;
yes
| ?- hates(_,pat).
no
| ?- hates(_,fox).
true ? ;
no
| ?-
```

```
var_anonymous.pl
1  hates(jim,tom).
2  hates(pat,bob).
3  hates(dog,fox).
4  hates(peter,tom).
```

Seccion 8.- Operators: muestra las distintas operaciones que se pueden realizar

The screenshot shows the GNU Prolog console on the left and a code editor on the right. The console displays the compilation of 'op_arith.pl' and the execution of the 'calc' predicate. The code editor shows the source code for 'op_arith.pl'.

```
GNU Prolog console
File Edit Terminal Prolog Help
GNU Prolog 1.5.0 (64 bits)
Compiled Jul  8 2021, 12:22:53 with gcc
Copyright (C) 1999-2021 Daniel Diaz

| ?- change_directory('C:/Users/PC/Desktop/4to/Paradigmas-de-prgramacion/Prolog').
yes
| ?- [op_arith].
compiling C:/Users/PC/Desktop/4to/Paradigmas-de-prgramacion/Prolog/op_arith.pl for byte code...
C:/Users/PC/Desktop/4to/Paradigmas-de-prgramacion/Prolog/op_arith.pl compiled, 6 lines read - 2452 bytes
yes
| ?- calc.
100 + 200 is 300
400 - 150 is 250
10 * 300 is 3000
100 / 30 is 3.3333333333333335
100 // 30 is 3
100 ** 2 is 10000.0
100 mod 30 is 10
yes
| ?-
```

```
op_arith.pl
1  calc :- X is 100 + 200,write('100 + 200 is '),write(X),nl,
2         Y is 400 - 150,write('400 - 150 is '),write(Y),nl,
3         Z is 10 * 300,write('10 * 300 is '),write(Z),nl,
4         A is 100 / 30,write('100 / 30 is '),write(A),nl,
5         B is 100 // 30,write('100 // 30 is '),write(B),nl,
6         C is 100 ** 2,write('100 ** 2 is '),write(C),nl,
7         D is 100 mod 30,write('100 mod 30 is '),write(D),nl.
```

Seccion 9.- Loop & Decision Making: Creacion de ciclos

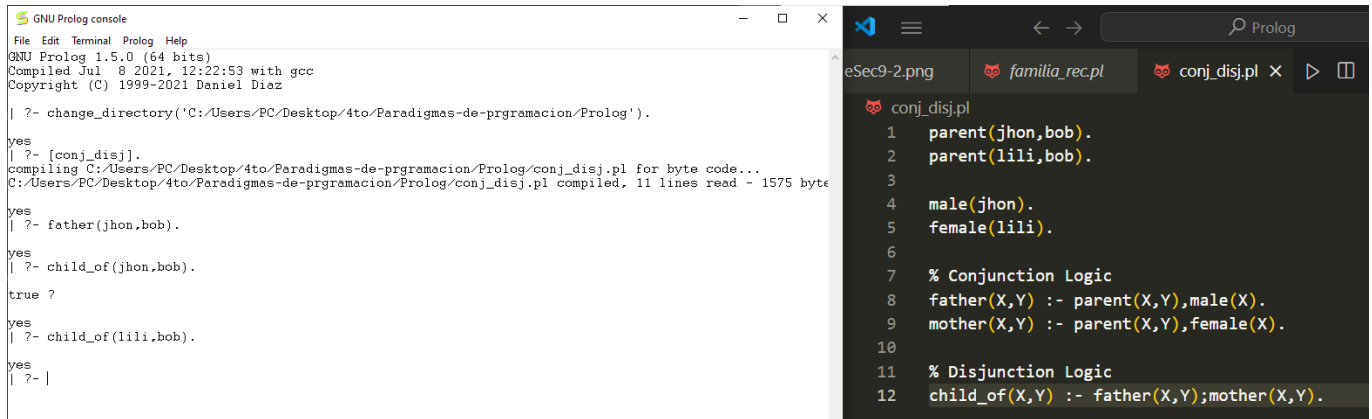
The screenshot shows the GNU Prolog console on the left and a code editor on the right. The console displays the compilation of 'test.pl' and the execution of the 'gt' and 'gte' predicates. The code editor shows the source code for 'test.pl'.

```
GNU Prolog console
File Edit Terminal Prolog Help
GNU Prolog 1.5.0 (64 bits)
Compiled Jul  8 2021, 12:22:53 with gcc
Copyright (C) 1999-2021 Daniel Diaz

| ?- change_directory('C:/Users/PC/Desktop/4to/Paradigmas-de-prgramacion/Prolog').
yes
| ?- [test].
compiling C:/Users/PC/Desktop/4to/Paradigmas-de-prgramacion/Prolog/test.pl for byte code...
C:/Users/PC/Desktop/4to/Paradigmas-de-prgramacion/Prolog/test.pl compiled, 9 lines read - 1156 bytes written
yes
| ?- gt(10,100).
X is smaller
yes
| ?- gt(150,100).
X is greater or equal
true ?
Action (; for next solution, a for all solutions, RET to stop) ?
yes
| ?- gte(10,20).
X is smaller
yes
| ?- gte(100,20).
X is greater
true ?
yes
| ?- gte(100,100).
X and Y are same
true ?
yes
| ?-
```

```
test.pl
1  % If-Then-Else statement
2
3  gt(X,Y) :- X >= Y,write('X is greater or equal').
4  gt(X,Y) :- X < Y,write('X is smaller').
5
6  % If-Elif-Else statement
7
8  gte(X,Y) :- X > Y,write('X is greater').
9  gte(X,Y) :- X == Y,write('X and Y are same').
10 gte(X,Y) :- X < Y,write('X is smaller').
```

Seccion 10.- Conjunctions & Disjunctions



```

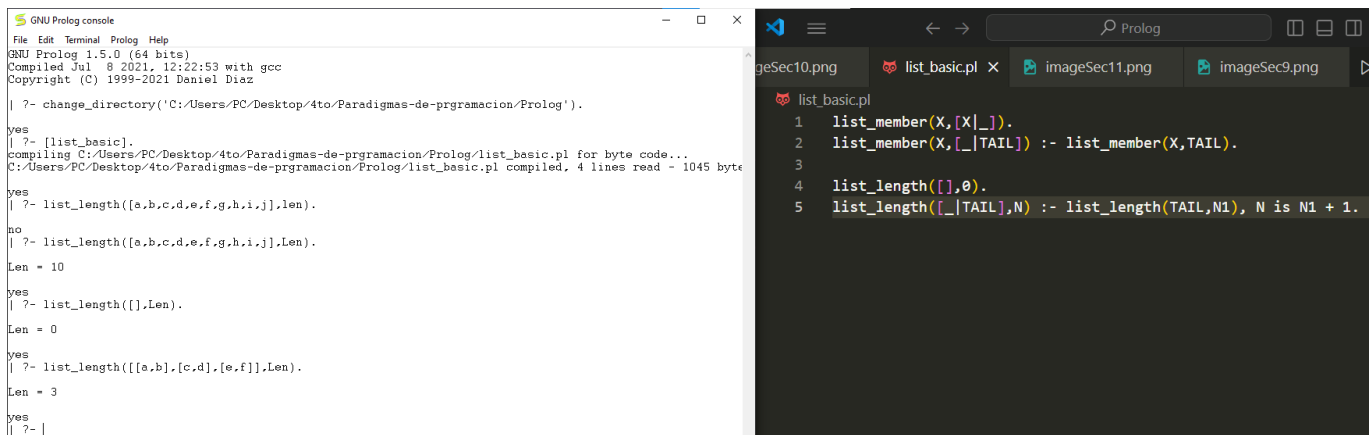
GNU Prolog console
File Edit Terminal Prolog Help
GNU Prolog 1.5.0 (64 bits)
Compiled Jul  8 2021, 12:22:53 with gcc
Copyright (C) 1999-2021 Daniel Diaz

| ?- change_directory('C:/Users/PC/Desktop/4to/Paradigmas-de-prgramacion/Prolog').
yes
| ?- [conj_disj].
compiling C:/Users/PC/Desktop/4to/Paradigmas-de-prgramacion/Prolog/conj_disj.pl for byte code...
C:/Users/PC/Desktop/4to/Paradigmas-de-prgramacion/Prolog/conj_disj.pl compiled, 11 lines read - 1575 byte
yes
| ?- father(jhon,bob).
yes
| ?- child_of(jhon,bob).
true ?
yes
| ?- child_of(lili,bob).
yes
| ?- |

conj_disj.pl
1  parent(jhon,bob).
2  parent(lili,bob).
3
4  male(jhon).
5  female(lili).
6
7  % Conjunction Logic
8  father(X,Y) :- parent(X,Y),male(X).
9  mother(X,Y) :- parent(X,Y),female(X).
10
11 % Disjunction Logic
12 child_of(X,Y) :- father(X,Y);mother(X,Y).

```

Section 11.- List: Presenta la creacion de listas y las ditintas acciones que se le pueden realizar a estas



```

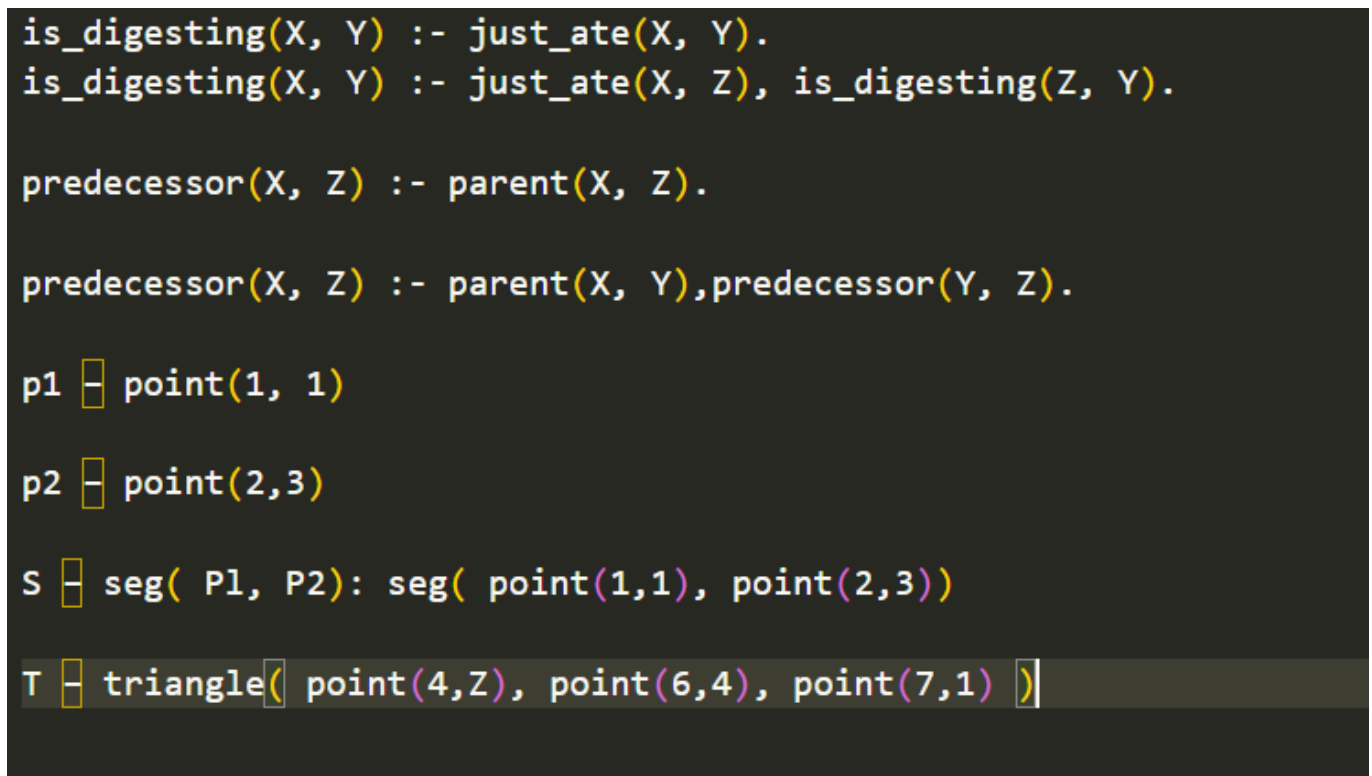
GNU Prolog console
File Edit Terminal Prolog Help
GNU Prolog 1.5.0 (64 bits)
Compiled Jul  8 2021, 12:22:53 with gcc
Copyright (C) 1999-2021 Daniel Diaz

| ?- change_directory('C:/Users/PC/Desktop/4to/Paradigmas-de-prgramacion/Prolog').
yes
| ?- [list_basic].
compiling C:/Users/PC/Desktop/4to/Paradigmas-de-prgramacion/Prolog/list_basic.pl for byte code...
C:/Users/PC/Desktop/4to/Paradigmas-de-prgramacion/Prolog/list_basic.pl compiled, 4 lines read - 1045 byte
yes
| ?- list_length([a,b,c,d,e,f,g,h,i,j],len).
no
| ?- list_length([a,b,c,d,e,f,g,h,i,j],Len).
Len = 10
yes
| ?- list_length([],Len).
Len = 0
yes
| ?- list_length([a,b],[c,d],[e,f],Len).
Len = 3
yes
| ?- |

list_basic.pl
1  list_member(X,[X|_]).
2  list_member(X,[_|TAIL]) :- list_member(X,TAIL).
3
4  list_length([],0).
5  list_length([_|TAIL],N) :- list_length(TAIL,N1), N is N1 + 1.

```

Section 12.- Recursion and Structures: Presenta la recursion y explica arboles



```

is_digesting(X, Y) :- just_ate(X, Y).
is_digesting(X, Y) :- just_ate(X, Z), is_digesting(Z, Y).

predecessor(X, Z) :- parent(X, Z).

predecessor(X, Z) :- parent(X, Y),predecessor(Y, Z).

p1 [ point(1, 1)
p2 [ point(2,3)
S [ seg( P1, P2): seg( point(1,1), point(2,3))
T [ triangle( point(4,Z), point(6,4), point(7,1) )

```

Section 13.- Backtracking



The image shows a screenshot of the GNU Prolog console window. The title bar reads 'GNU Prolog console'. The menu bar includes 'File', 'Edit', 'Terminal', 'Prolog', and 'Help'. The main text area contains the following output:

```
GNU Prolog 1.5.0 (64 bits)
Compiled Jul  8 2021, 12:22:53 with gcc
Copyright (C) 1999-2021 Daniel Diaz

| ?- change_directory('C:/Users/PC/Desktop/4to/Paradigmas-de-prgramacion/Prolog').

yes
| ?- [back_track].
compiling C:/Users/PC/Desktop/4to/Paradigmas-de-prgramacion/Prolog/back_track.pl for byte code...
C:/Users/PC/Desktop/4to/Paradigmas-de-prgramacion/Prolog/back_track.pl compiled, 26 lines read - 2886 byt

yes
| ?- likes(mary,elephant).

yes
| ?- likes(mary,tiger).

yes
| ?- likes(mary,python).

no
| ?- likes(mary,cobra).

no
| ?- trace.
The debugger will first creep -- showing everything (trace)

yes
{trace}
| ?- likes(mary,dog).
    1    1  Call: likes(mary,dog) ?
    2    2  Call: snake(dog) ?
    2    2  Fail: snake(dog) ?
    2    2  Call: animal(dog) ?
    2    2  Exit: animal(dog) ?
    1    1  Exit: likes(mary,dog) ?

yes
{trace}
| ?- likes(mary,python).
    1    1  Call: likes(mary,python) ?
    2    2  Call: snake(python) ?
    2    2  Exit: snake(python) ?
    3    2  Call: fail ?
    3    2  Fail: fail ?
    1    1  Fail: likes(mary,python) ?

no
{trace}
| ?- S
```

Seccion 14.- Different and Not: Presenta los predicados de diferente y no, que comprueban si 2 argumetos son iguales o no

```

GNU Prolog console
File Edit Terminal Prolog Help
GNU Prolog 1.5.0 (64 bits)
Compiled Jul  8 2021, 12:22:53 with gcc
Copyright (C) 1999-2021 Daniel Diaz

?- change_directory('C:/Users/PC/Desktop/4to/Paradigmas-de-prgramacion/Prolog').

yes
?- [diff_rel].
compiling C:/Users/PC/Desktop/4to/Paradigmas-de-prgramacion/Prolog/diff_rel.pl for byte code...
!:/Users/PC/Desktop/4to/Paradigmas-de-prgramacion/Prolog/diff_rel.pl:2: warning: singleton variables |
!:/Users/PC/Desktop/4to/Paradigmas-de-prgramacion/Prolog/diff_rel.pl compiled, 3 lines read - 751 bytes

yes
?- different(100,200).

true ?

yes
?- different(100,100).

no
?- different(abc,def).

true ?

yes
?- different(abc,abc).

no
?- |

```

Seccion 15.- Intputs and Outputs: Presenta la escritura y lectura de archivos

```

GNU Prolog console
File Edit Terminal Prolog Help
GNU Prolog 1.5.0 (64 bits)
Compiled Jul  8 2021, 12:22:53 with gcc
Copyright (C) 1999-2021 Daniel Diaz

| ?- change_directory('C:/Users/PC/Desktop/4to/Paradigmas-de-prgramacion/Prolog').

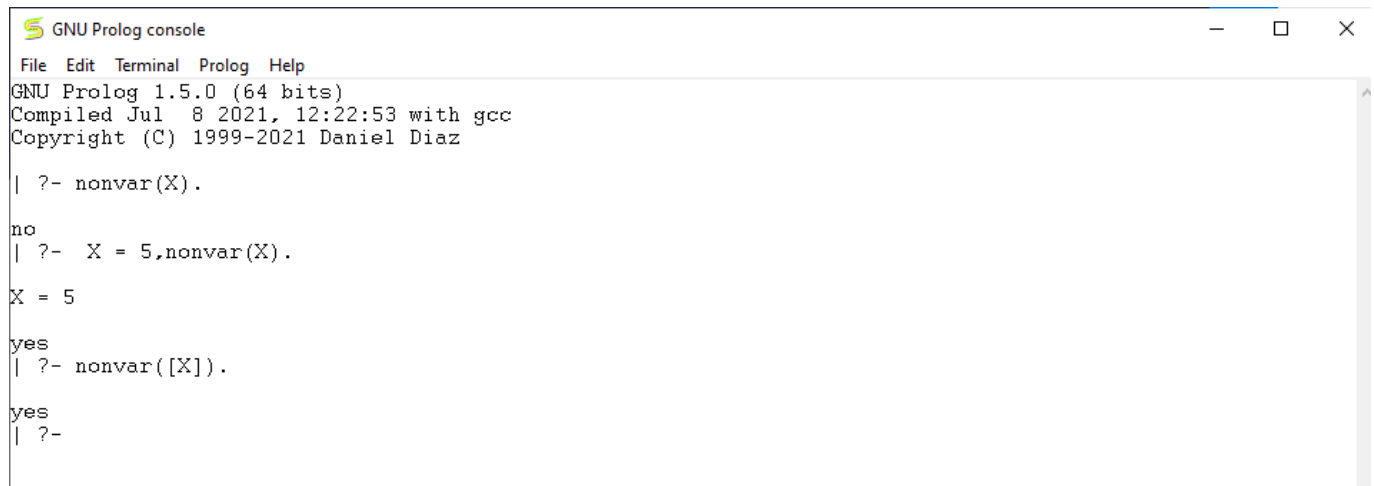
yes
| ?- [read_write].
compiling C:/Users/PC/Desktop/4to/Paradigmas-de-prgramacion/Prolog/read_write.pl for byte code...
C:/Users/PC/Desktop/4to/Paradigmas-de-prgramacion/Prolog/read_write.pl compiled, 7 lines read - 1288 bytes

yes
| ?- cube.
Write a number: 2
.
Cube of 2: 8
Write a number: 10.
Cube of 10: 1000
Write a number: 12.
Cube of 12: 1728
Write a number: 8.
Cube of 8: 512
Write a number: stop
.

(32 ms) yes
| ?- |

```

Seccion 16.- Built-In Predicates: Predicados integrados en Prolog



```
GNU Prolog console
File Edit Terminal Prolog Help
GNU Prolog 1.5.0 (64 bits)
Compiled Jul  8 2021, 12:22:53 with gcc
Copyright (C) 1999-2021 Daniel Diaz

| ?- nonvar(X).
no
| ?- X = 5,nonvar(X).
X = 5
yes
| ?- nonvar([X]).
yes
| ?-
```

Seccion 17.- Tree Data Structure: Se presenta la manera de implementar una estructura de arbol

```

GNU Prolog console
File Edit Terminal Prolog Help
GNU Prolog 1.5.0 (64 bits)
Compiled Jul  8 2021, 12:22:53 with gcc
Copyright (C) 1999-2021 Daniel Diaz

| ?- change_directory('C:/Users/PC/Desktop/4to/Paradigmas-de-prgramacion/Prolog
yes
| ?- [case_tree].
compiling C:/Users/PC/Desktop/4to/Paradigmas-de-prgramacion/Prolog/case_tree.p
C:/Users/PC/Desktop/4to/Paradigmas-de-prgramacion/Prolog/case_tree.pl:17: warn
C:/Users/PC/Desktop/4to/Paradigmas-de-prgramacion/Prolog/case_tree.pl compiled
yes
| ?- i is_parent p.
yes
| ?- i is_parent s.
no
| ?- is_parent(i,p).
yes
| ?- e is_sibling_of f.
true ? |
yes
| ?- is_sibling_of(e,g).
no
| ?- leaf_node(v).
yes
| ?- leaf_node(a).
no
| ?- is_at_same_level(l,s).
true ? ;
no
| ?- l is_at_same_level v.
no
| ?-

```

Conclusion:

La programacion logica me proporciono un enfoque distinto a la hora de programar y en la resolucion de problemas, ofreciendo maneras intuitivas para lograrlo. Aunque es algo dificil de entender y por lo visto tiene areas de aplicacion muy especifica como la IA, procesamiento de datos, entre otros.