

II. PROGRAMACION LOGICA

El lenguaje de programación PROLOG ("PROgrammation en LOGique") fue creado por **Alain Colmerauer** y sus colaboradores alrededor de 1970 en la **Universidad de Marseille-Aix**, si bien uno de los principales protagonistas de su desarrollo y promoción fue **Robert Kowalski** de la **Universidad de Edimburgh**. Las investigaciones de **Kowalski** proporcionaron el marco teórico, mientras que los trabajos de **Colmerauer** dieron origen al actual lenguaje de programación, construyendo el primer interprete Prolog. **David Warren**, de la **Universidad de Edimburgh**, desarrolló el primer compilador de Prolog (WAM – "Warren Abstract Machine"). Se pretendía usar la lógica formal como base para un lenguaje de programación, es decir, era un primer intento de diseñar un lenguaje de programación que posibilitara al programador especificar sus problemas en lógica. Lo que lo diferencia de los demás es el énfasis sobre la especificación del problema. Es un lenguaje para el procesamiento de información simbólica. PROLOG es una realización aproximada del modelo de computación de Programación Lógica sobre una máquina secuencial. Desde luego, no es la única realización posible, pero sí es la mejor elección práctica, ya que equilibra por un lado la preservación de las propiedades del modelo abstracto de Programación Lógica y por el otro lado consigue que la implementación sea eficiente.

El lenguaje PROLOG juega un importante papel dentro de la Inteligencia Artificial, y se propuso como el lenguaje nativo de las máquinas de la quinta generación ("Fifth Generation Kernel Language", FGKL) que quería que fueran Sistemas de Procesamiento de Conocimiento. La expansión y el uso de este lenguaje propició la aparición de la normalización del lenguaje Prolog con la norma ISO (propuesta de Junio de 1993).

PROLOG es un lenguaje de programación para ordenadores que se basa en el lenguaje de la Lógica de Primer Orden y que se utiliza para resolver problemas en los que entran en juego objetos y relaciones entre ellos. Por ejemplo, cuando decimos "Jorge tiene una moto", estamos expresando una relación entre un objeto (Jorge) y otro objeto en particular (una moto). Más aún, estas relaciones tienen un orden específico (Jorge posee la moto y no al contrario). Por otra parte, cuando realizamos una pregunta (¿Tiene Jorge una moto?) lo que estamos haciendo es indagando acerca de una relación.

Además, también solemos usar reglas para describir relaciones: "dos personas son hermanas si ambas son hembras y tienen los mismos padres". Como veremos más adelante, esto es lo que hacemos en Prolog.

Una de las ventajas de la programación lógica es que se especifica qué se tiene que hacer (**programación declarativa**), y no cómo se debe hacer (**programación imperativa**). A pesar de esto, Prolog incluye algunos predicados predefinidos metalógicos, ajenos al ámbito de la Lógica de Primer Orden, (var, nonvar, ==, ...), otros extra-lógicos, que tienen un efecto lateral, (write, get, ...) y un tercer grupo que nos sirven para expresar información de control de como realizar alguna tarea (el corte, ...).

Por tanto, Prolog ofrece un sistema de programación práctico que tiene algunas de las ventajas de claridad y declaratividad que ofrecería un lenguaje de programación lógica y, al mismo tiempo, nos permite un cierto control y operatividad.



Estructura general de un Programa en Visual Prolog

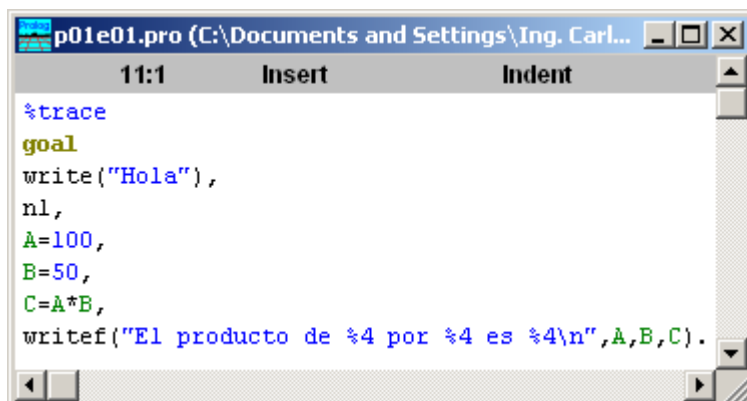
Un programa de Visual Prolog consiste en conjunto de secciones programables. Cada sección del programa se identifica por una palabra clave, como se muestra a continuación:

Sección	Contenido
Compiler directives	Opciones al inicio del programa.
CONSTANTS section	Cero o más Constantes.
DOMAINS section	Cero o más declaraciones de dominio.
DATABASE section	Cero o más predicados de la Base de Datos.
PREDICATES section	Cero o más declaraciones de Predicado.
GOAL section	Construir en (interior) la meta (Resultados).
CLAUSES section	Cero o más Cláusulas.

Un programa puede contener varios **domains**, **predicates**, **database** o **clauses**, con tal de que usted observe las restricciones siguientes:

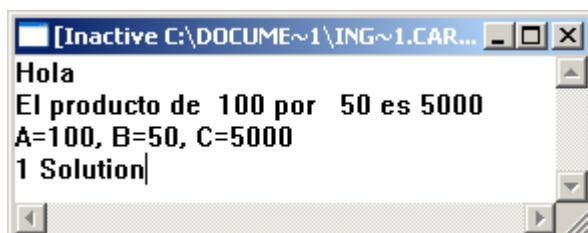
- Las Opciones deben preceder las otras secciones
- Deben definirse constantes, dominios, y predicados antes de que usted los use. Sin embargo, dentro de la sección de los DOMINIOS usted puede referirse a dominios que se declaran más tarde.
- Sólo una meta (goal) debe reunirse durante la recopilación. Sin embargo, la meta puede aparecer en cualquier parte después de la sección de los PREDICADOS declarando su sub metas.
- Todas las cláusulas que describen el mismo predicado deben ocurrir en la sucesión (uno después del otro).
- Todas las declaraciones globales deben venir antes de cualquier declaración local.
- Las secciones de Base de Datos pueden nombrarse, pero un nombre dado sólo puede aparecer una vez.

1. Dados dos números, imprimir su producto.



```
%trace
goal
write("Hola"),
nl,
A=100,
B=50,
C=A*B,
writef("El producto de %4 por %4 es %4\n",A,B,C).
```

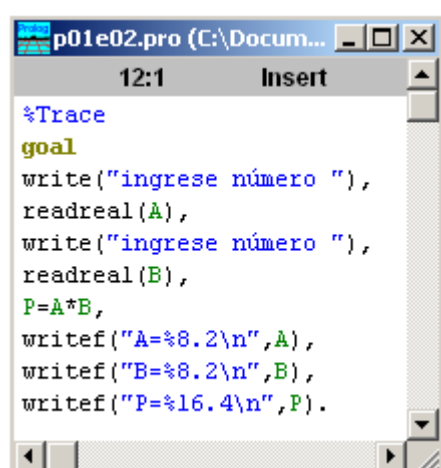
Solución:



```
Hola
El producto de 100 por 50 es 5000
A=100, B=50, C=5000
1 Solution
```



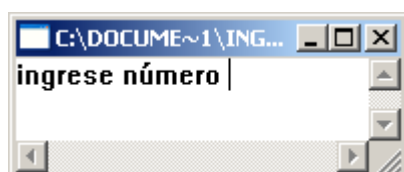
2. Dado dos números ingresados por teclado, imprimir su producto.



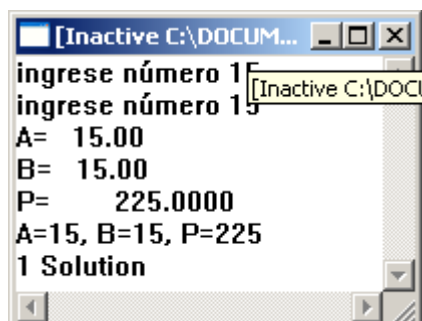
```
%Trace
goal
write("ingrese número "),
readreal(A),
write("ingrese número "),
readreal(B),
P=A*B,
writef("A=%8.2\n",A),
writef("B=%8.2\n",B),
writef("P=%16.4\n",P).
```

Solución:

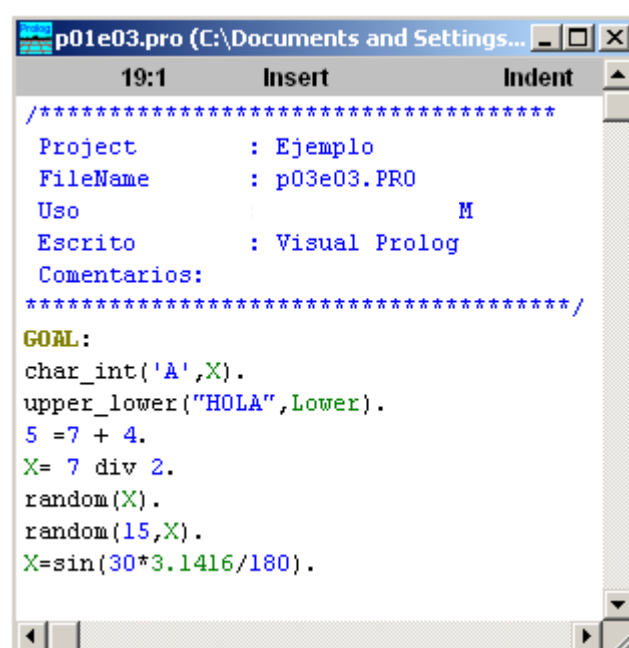
Paso 1: Ingreso de Datos



Paso 2: Muestra de Resultados

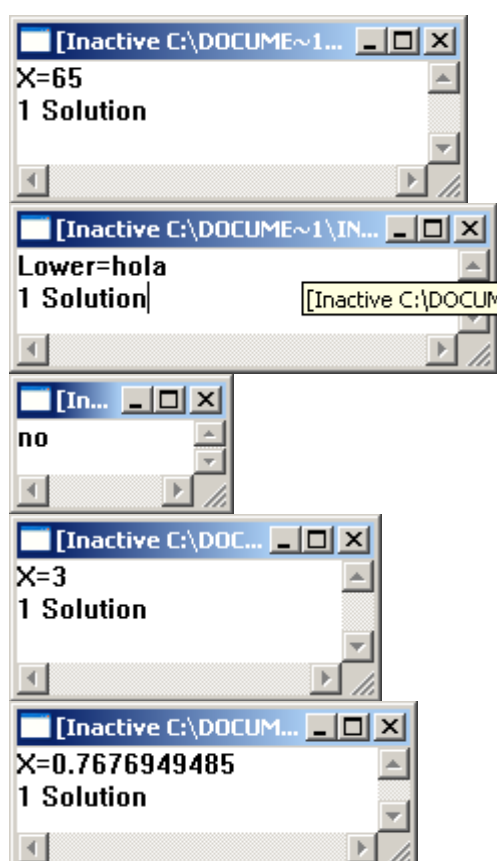


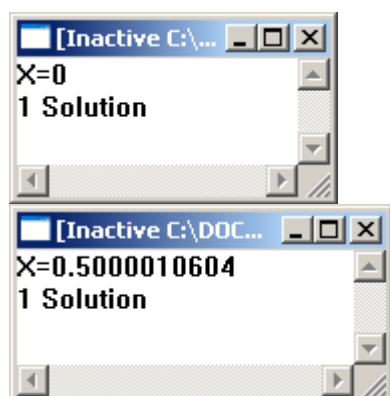
3. Estructuras básicas de datos.



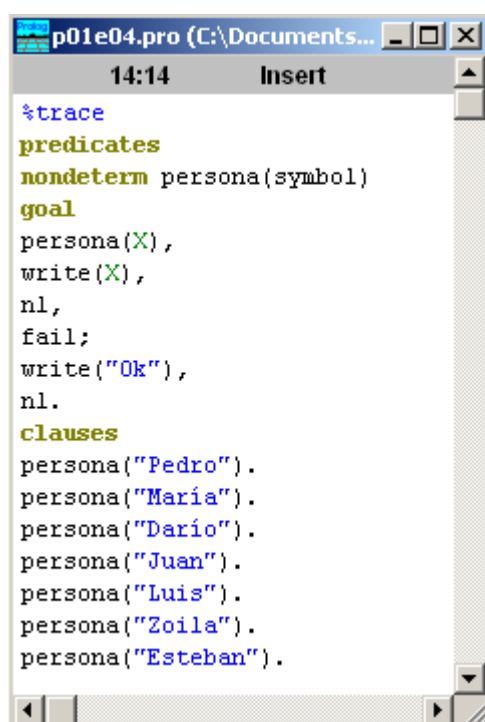
```
p01e03.pro (C:\Documents and Settings...  
19:1      Insert      Indent  
/*****  
Project      : Ejemplo  
FileName     : p03e03.PRO  
Uso          : M  
Escrito      : Visual Prolog  
Comentarios:  
*****/  
GOAL:  
char_int('A',X).  
upper_lower("HOLA",Lower).  
5 = 7 + 4.  
X= 7 div 2.  
random(X).  
random(15,X).  
X=sin(30*3.1416/180).
```

Soluciones

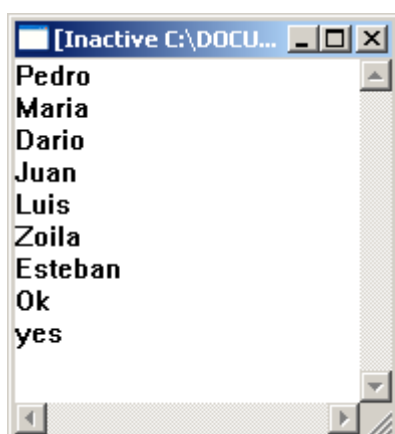




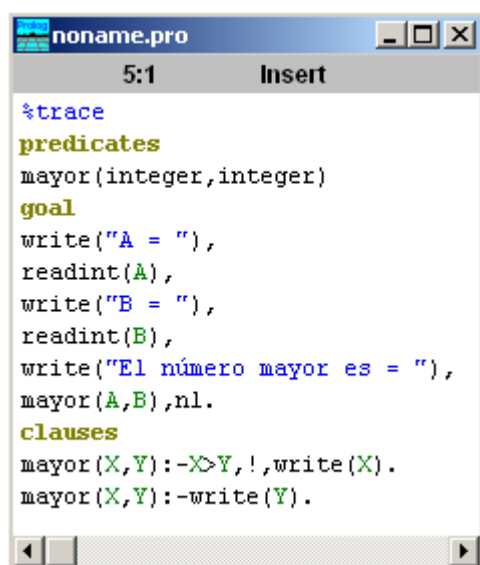
4. Teniendo en cuenta cláusulas, imprimirlas.



Solución:

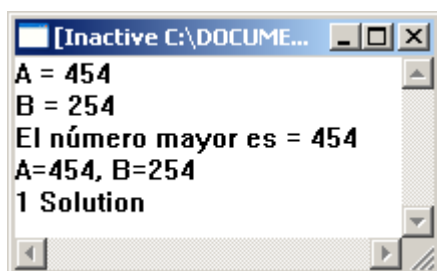


5. Ingresar dos números por teclado y decir cuál de ellos es el mayor.



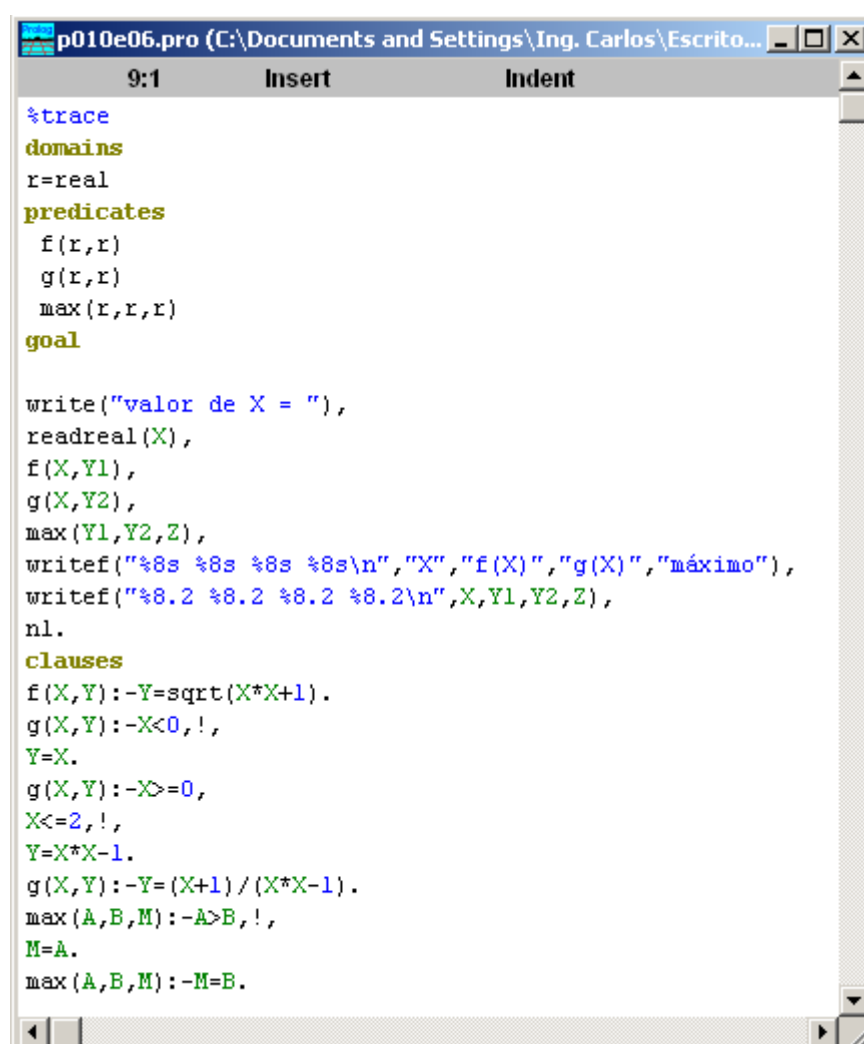
```
%trace
predicates
mayor(integer,integer)
goal
write("A = "),
readint(A),
write("B = "),
readint(B),
write("El número mayor es = "),
mayor(A,B),nl.
clauses
mayor(X,Y):-X>Y,!,write(X).
mayor(X,Y):-write(Y).
```

Solución:



```
A = 454
B = 254
El número mayor es = 454
A=454, B=254
1 Solution
```

6. Resolver una función polinómica conociendo el valor de la variable.

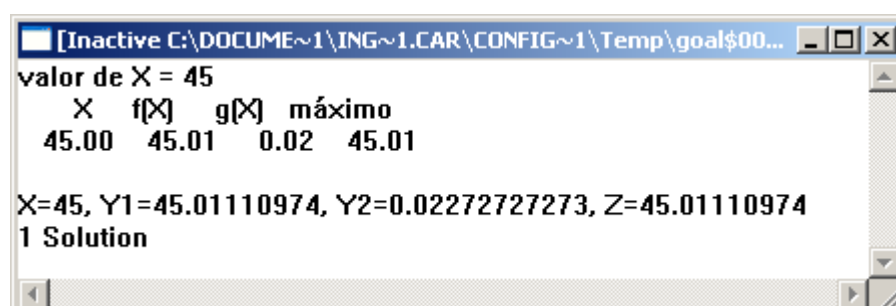


```
p010e06.pro (C:\Documents and Settings\Ing. Carlos\Escrito...
9:1      Insert      Indent

%trace
domains
r=real
predicates
  f(r,r)
  g(r,r)
  max(r,r,r)
goal

write("valor de X = "),
readreal(X),
f(X,Y1),
g(X,Y2),
max(Y1,Y2,Z),
writef("%8s %8s %8s %8s\n","X","f(X)","g(X)","máximo"),
writef("%8.2 %8.2 %8.2 %8.2\n",X,Y1,Y2,Z),
nl.
clauses
f(X,Y):-Y=sqrt(X*X+1).
g(X,Y):-X<0,! ,
Y=X.
g(X,Y):-X>=0,
X<=2,! ,
Y=X*X-1.
g(X,Y):-Y=(X+1)/(X*X-1).
max(A,B,M):-A>B,! ,
M=A.
max(A,B,M):-M=B.
```

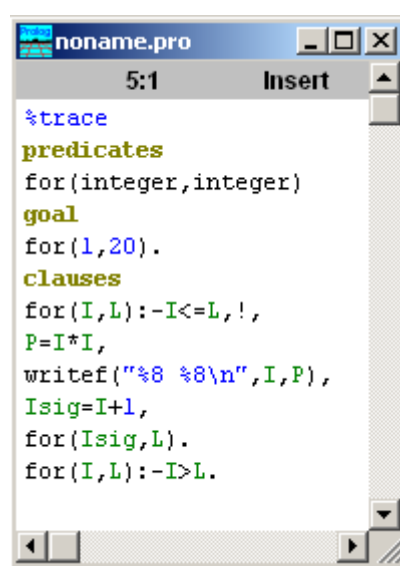
Solución:



```
[Inactive C:\DOCUME~1\ING~1\CAR\CONFIG~1\Temp\goal$00...
valor de X = 45
  X  f(X)  g(X)  máximo
45.00 45.01  0.02 45.01

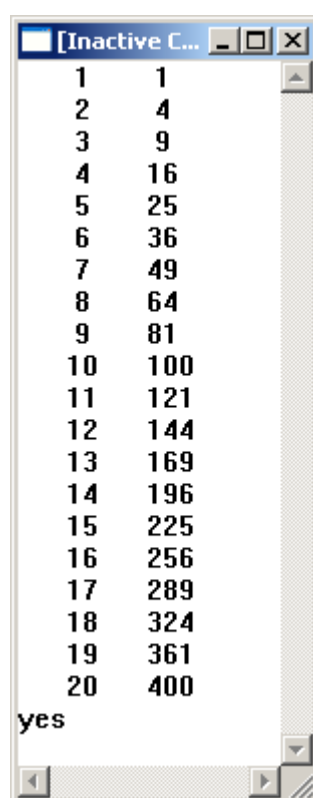
X=45, Y1=45.01110974, Y2=0.02272727273, Z=45.01110974
1 Solution
```

7. Imprimir las primeras 20 potencias con su indicador.



```
%trace
predicates
for(integer,integer)
goal
for(1,20).
clauses
for(I,L):-I<=L,!,
P=I*I,
writef("%8 %8\n",I,P),
Isig=I+1,
for(Isig,L).
for(I,L):-I>L.
```

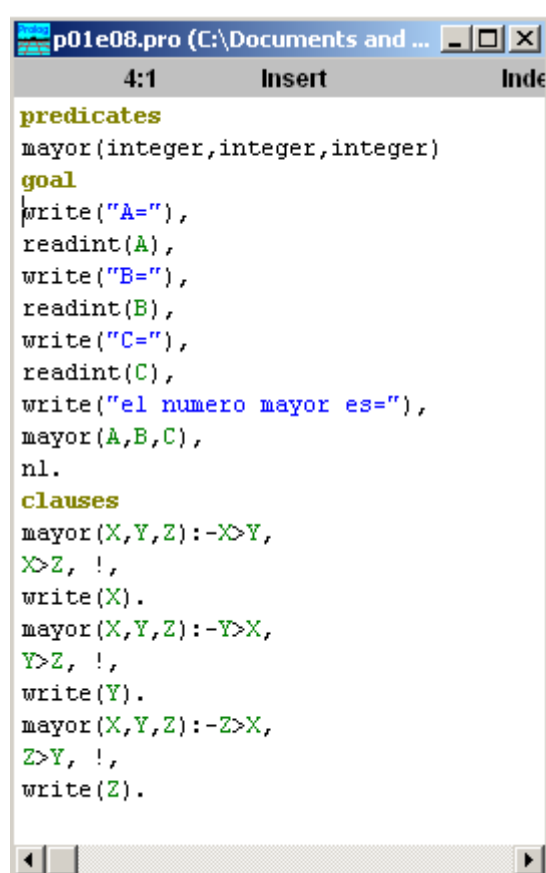
Solución:



1	1
2	4
3	9
4	16
5	25
6	36
7	49
8	64
9	81
10	100
11	121
12	144
13	169
14	196
15	225
16	256
17	289
18	324
19	361
20	400

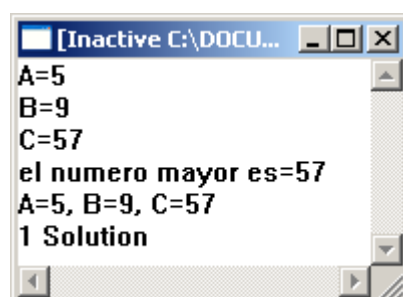
yes

8. Escriba un programa que lea tres números enteros e imprima el mayor.



```
p01e08.pro (C:\Documents and ...  
4:1      Insert      Inde  
  
predicates  
mayor(integer,integer,integer)  
goal  
|write("A="),  
|readint(A),  
|write("B="),  
|readint(B),  
|write("C="),  
|readint(C),  
|write("el numero mayor es="),  
|mayor(A,B,C),  
|nl.  
clauses  
mayor(X,Y,Z):-X>Y,  
X>Z, !,  
write(X).  
mayor(X,Y,Z):-Y>X,  
Y>Z, !,  
write(Y).  
mayor(X,Y,Z):-Z>X,  
Z>Y, !,  
write(Z).
```

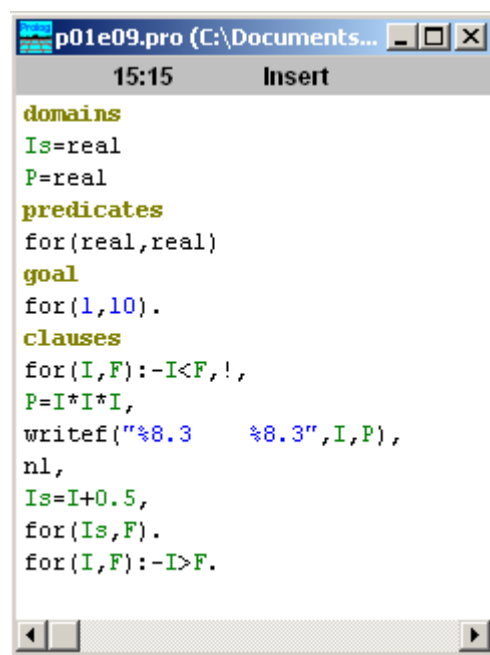
Solución:



```
[Inactive C:\DOCU...  
A=5  
B=9  
C=57  
el numero mayor es=57  
A=5, B=9, C=57  
1 Solution
```

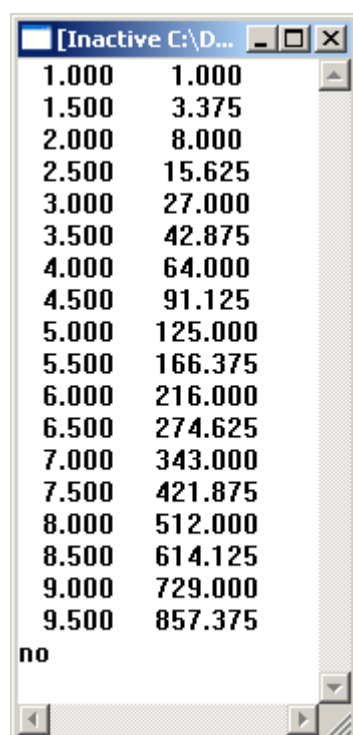
9. Escriba un programa que imprima la tabla de los números del 1.0, 1.5, ..., 9.5, 10.0 y sus respectivos cubos.

1.0	1.000
1.5	3.375



```
domains
Is=real
P=real
predicates
for(real,real)
goal
for(1,10).
clauses
for(I,F):-I<F,!,
P=I*I*I,
writef("%8.3    %8.3",I,P),
nl,
Is=I+0.5,
for(Is,F).
for(I,F):-I>F.
```

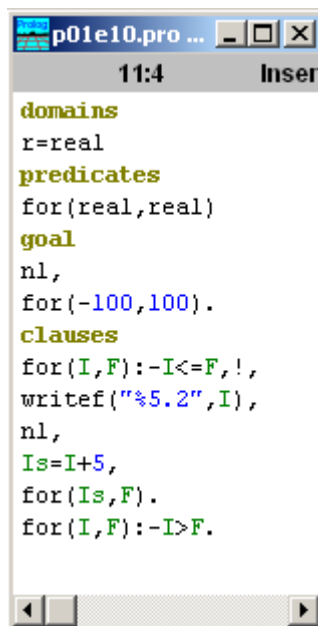
Solución:



1.000	1.000
1.500	3.375
2.000	8.000
2.500	15.625
3.000	27.000
3.500	42.875
4.000	64.000
4.500	91.125
5.000	125.000
5.500	166.375
6.000	216.000
6.500	274.625
7.000	343.000
7.500	421.875
8.000	512.000
8.500	614.125
9.000	729.000
9.500	857.375

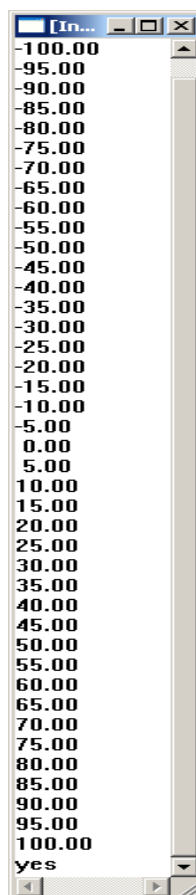
no

10. Escriba un programa que defina una función determinada.



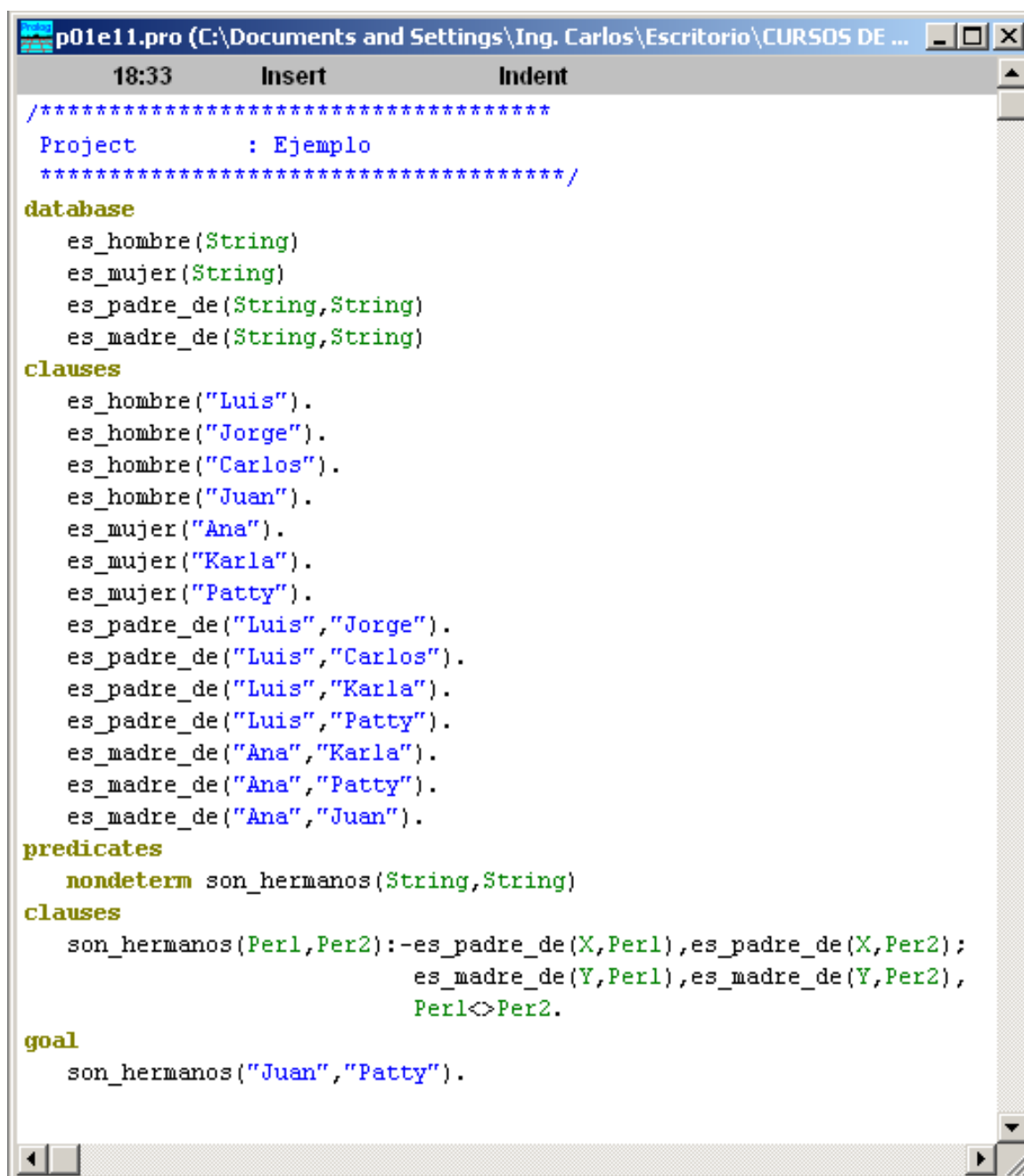
```
domains
r=real
predicates
for(real,real)
goal
nl,
for(-100,100).
clauses
for(I,F):-I<=F,!,
writef("%5.2",I),
nl,
Is=I+5,
for(Is,F).
for(I,F):-I>F.
```

Solución:



```
-100.00
-95.00
-90.00
-85.00
-80.00
-75.00
-70.00
-65.00
-60.00
-55.00
-50.00
-45.00
-40.00
-35.00
-30.00
-25.00
-20.00
-15.00
-10.00
-5.00
0.00
5.00
10.00
15.00
20.00
25.00
30.00
35.00
40.00
45.00
50.00
55.00
60.00
65.00
70.00
75.00
80.00
85.00
90.00
95.00
100.00
yes
```

11. Definición básica de relaciones de datos y cláusulas.



```
p01e11.pro (C:\Documents and Settings\Ing. Carlos\Escritorio\CURSOS DE ... 18:33 Insert Indent)

/*****
Project      : Ejemplo
*****/

database
    es_hombre(String)
    es_mujer(String)
    es_padre_de(String,String)
    es_madre_de(String,String)

clauses
    es_hombre("Luis").
    es_hombre("Jorge").
    es_hombre("Carlos").
    es_hombre("Juan").
    es_mujer("Ana").
    es_mujer("Karla").
    es_mujer("Patty").
    es_padre_de("Luis","Jorge").
    es_padre_de("Luis","Carlos").
    es_padre_de("Luis","Karla").
    es_padre_de("Luis","Patty").
    es_madre_de("Ana","Karla").
    es_madre_de("Ana","Patty").
    es_madre_de("Ana","Juan").

predicates
    nondeterm son_hermanos(String,String)

clauses
    son_hermanos(Per1,Per2):-es_padre_de(X,Per1),es_padre_de(X,Per2);
                           es_madre_de(Y,Per1),es_madre_de(Y,Per2),
                           Per1<>Per2.

goal
    son_hermanos("Juan","Patty").
```

Solución:

