

# EJEMPLO PROLOG

jafh

## EJEMPLO

Suponga que necesitamos tomar decisiones sobre las habilidades de programación de un grupo de personas para asignarles proyectos según esas habilidades.

# REPRESENTACIÓN

Aprobó el curso de...

Habilidad de ...

Para una mejor explicación manejemos el problema con diagramas, manejamos 2 cuadros el rosa muestra cursos que el alumno debe aprobar, el azul habilidades que el alumno adquirió como resultados de sus cursos.

# SE TIENEN LOS CURSOS

Programación I

Programación II

Estructura de datos

Programación OO

Programación VIS

Base de datos

# LAS HABILIDADES

Programador de  
Aplicaciones de  
gestión de datos

Programador  
básico

Programador de  
Aplicaciones OO

Programador de  
Aplicaciones  
estructuradas

# REGLAS

Una regla es una afirmación que puede ser cierta o falsa en dependencia de uno o varios hechos.

Por ejemplo decimos que un alumno tiene la habilidad de programador básico si aprobó el curso de programación I

Con un diagrama es:

DECIMOS QUE:

Programación I

Programador  
básico

Si el alumno aprobó los cursos de

Entonces tiene la habilidad de

## DECIMOS QUE:

Estructura de datos

Programación I

Programación II

Programador de  
Aplicaciones  
estructuradas

Si el alumno aprobó los cursos de

Entonces tiene la habilidad de



# REGLAS

Programador de  
Aplicaciones  
estructuradas

Programación OO

Programador de  
Aplicaciones OO

Si el alumno aprobó el cursos de POO  
y es programador de aplicaciones  
estructuradas

Entonces tiene la habilidad de

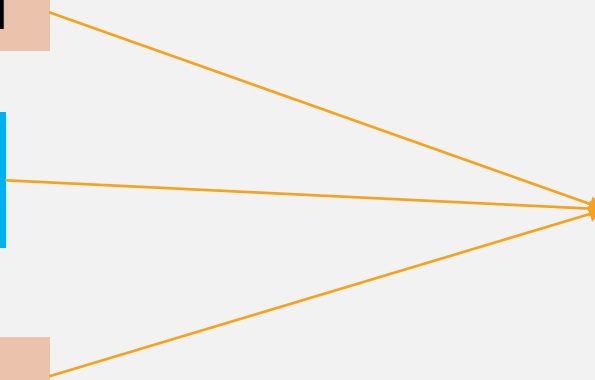
# REGLAS

Programación Visual

Programador de  
Aplicaciones OO

Base de datos

Programador de  
Aplicaciones de  
gestión de datos



# HECHOS

Son un conjunto de cosas que sabemos que se cumplen, por ejemplo suponga que sabemos que la alumna ana aprobó programación I, entonces un echo sería:

Ana aprobó programación I.

# REPRESENTACIÓN

Representemos ahora la información en Swi Prolog

# BASE DE CONOCIMIENTO

Suponga que sabemos los siguientes hechos:

## Representación en Prolog

```
aprobo(ana,programacion1). /*ana aprobó programación 1*/
```

```
aprobo(luis,programacion1).
```

```
aprobo(luis,programacion2).
```

```
aprobo(luis,estructuraDatos).
```

```
aprobo(luis,poo).
```

```
reprobo(rosa,programacion1). /* rosa reprobó programacion 1*/
```

```
reprobo(ana,programacion2).
```

# BASE DE CONOCIMIENTO

Suponga que tenemos las siguientes reglas:

## Representación en Prolog

*/\*X es programador básico si X aprobo programación 1\*/*  
`programadorBasico(X):-aprobo(X, programacion1).`

*/\*X es programador de aplicaciones estucturadas si X aprobo programación 1 programación 2 y estructura de datos\*/*  
`programadorAplicacionesEstructuradas(X):-aprobo(X, programacion1),aprobo(X, programacion2),aprobo(X, estructuraDatos).`

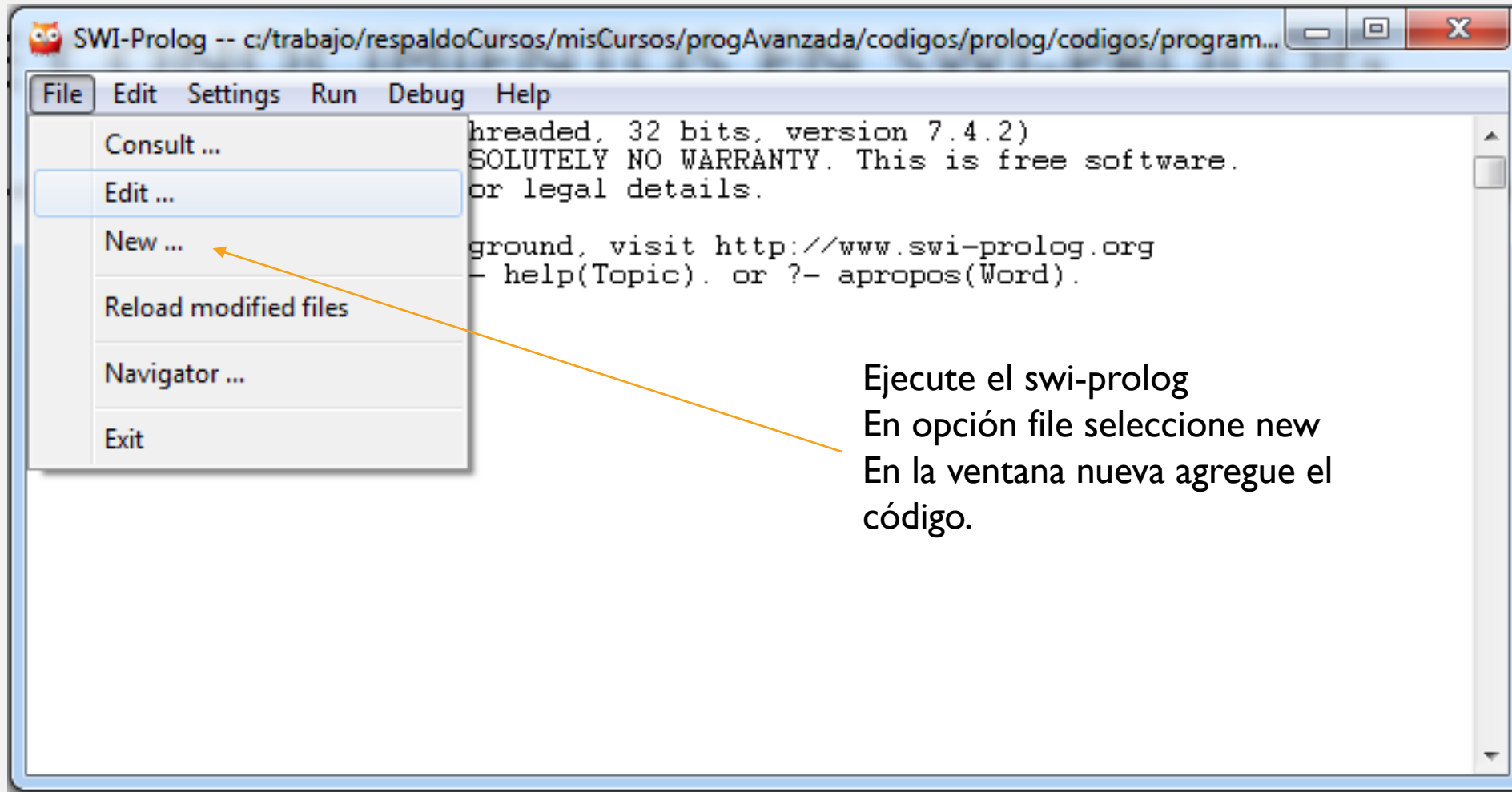
*/\*X es programador OO si X es programador de aplicaciones estructuradas y aprobo POO\*/*  
`programadorAplicacionesOO(X):-programadorAplicacionesEstructuradas(X),aprobo(X,poo).`

Nota: en la sentencia:

`programadorAplicacionesEstructuradas(X):-aprobo(X, programacion1),aprobo(X, programacion2),aprobo(X, estructuraDatos).`

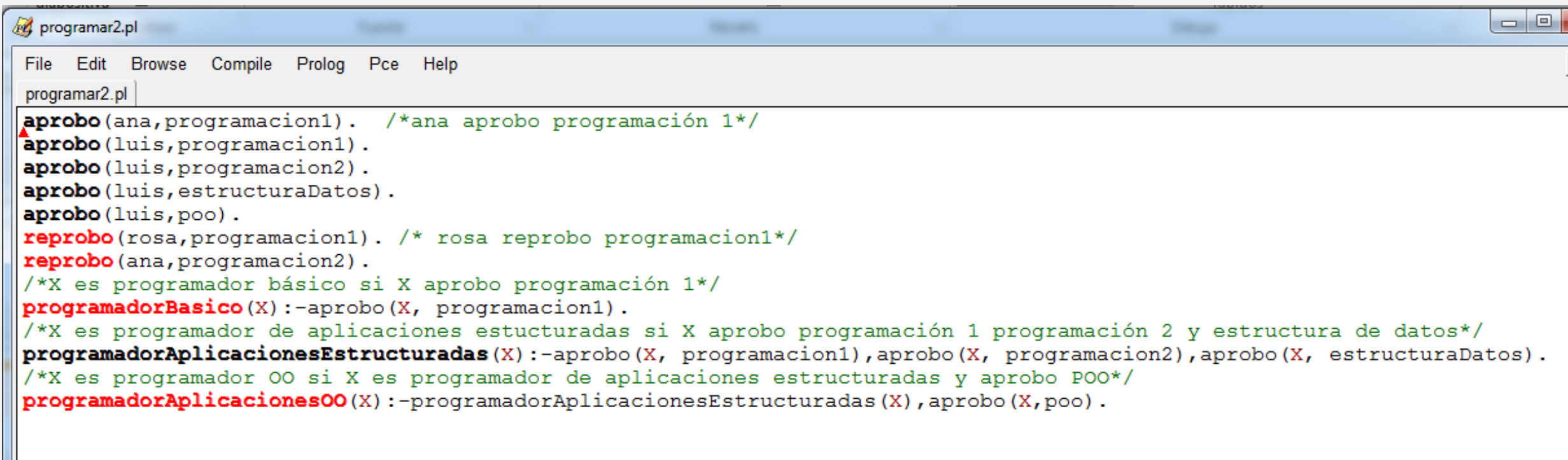
La coma representa and (; representa or)

# EDITE LA BASE DE CONOCIMIENTOS EN SWI-PROLOG



Ejecute el swi-prolog  
En opción file seleccione new  
En la ventana nueva agregue el  
código.

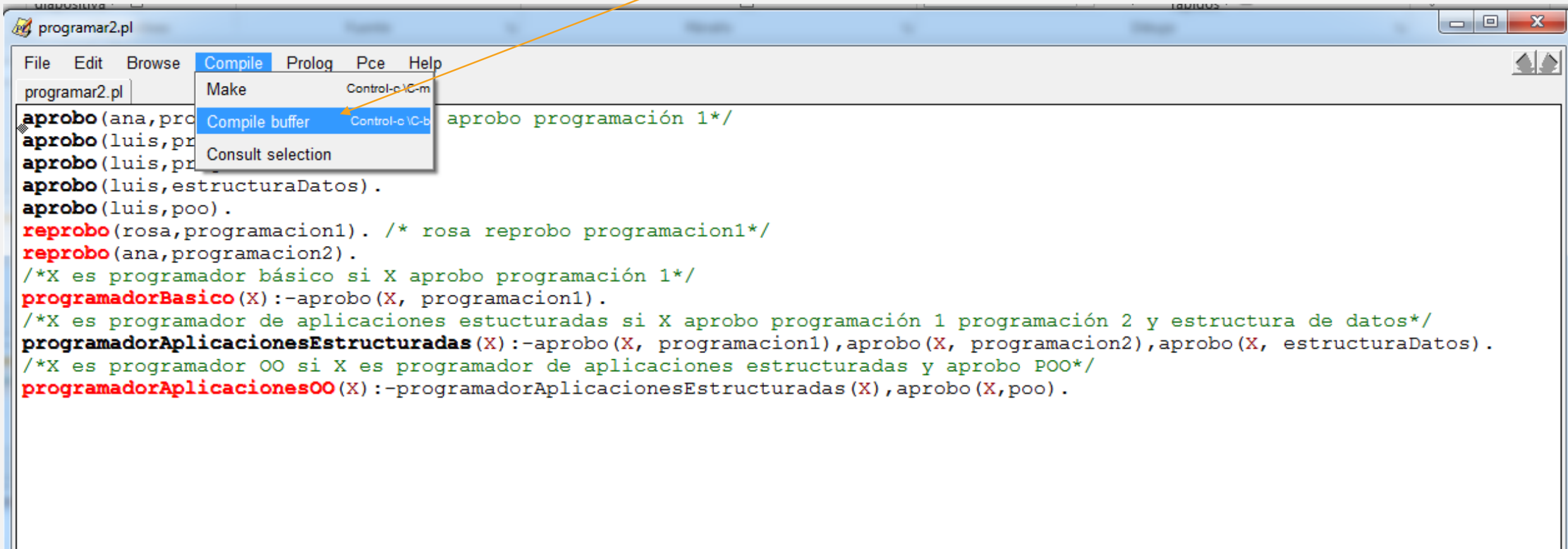
# AGREGUE EL CÓDIGO DE LA BASE DE C.



```
programar2.pl
File Edit Browse Compile Prolog Pce Help
programar2.pl
aprobo(ana,programacion1). /*ana aprobo programación 1*/
aprobo(luis,programacion1).
aprobo(luis,programacion2).
aprobo(luis,estructuraDatos).
aprobo(luis,poo).
reprobo(rosa,programacion1). /* rosa reprobo programacion1*/
reprobo(ana,programacion2).
/*X es programador básico si X aprobo programación 1*/
programadorBasico(X):-aprobo(X, programacion1).
/*X es programador de aplicaciones estructuradas si X aprobo programación 1 programación 2 y estructura de datos*/
programadorAplicacionesEstructuradas(X):-aprobo(X, programacion1),aprobo(X, programacion2),aprobo(X, estructuraDatos).
/*X es programador OO si X es programador de aplicaciones estructuradas y aprobo POO*/
programadorAplicacionesOO(X):-programadorAplicacionesEstructuradas(X),aprobo(X,poo).
```

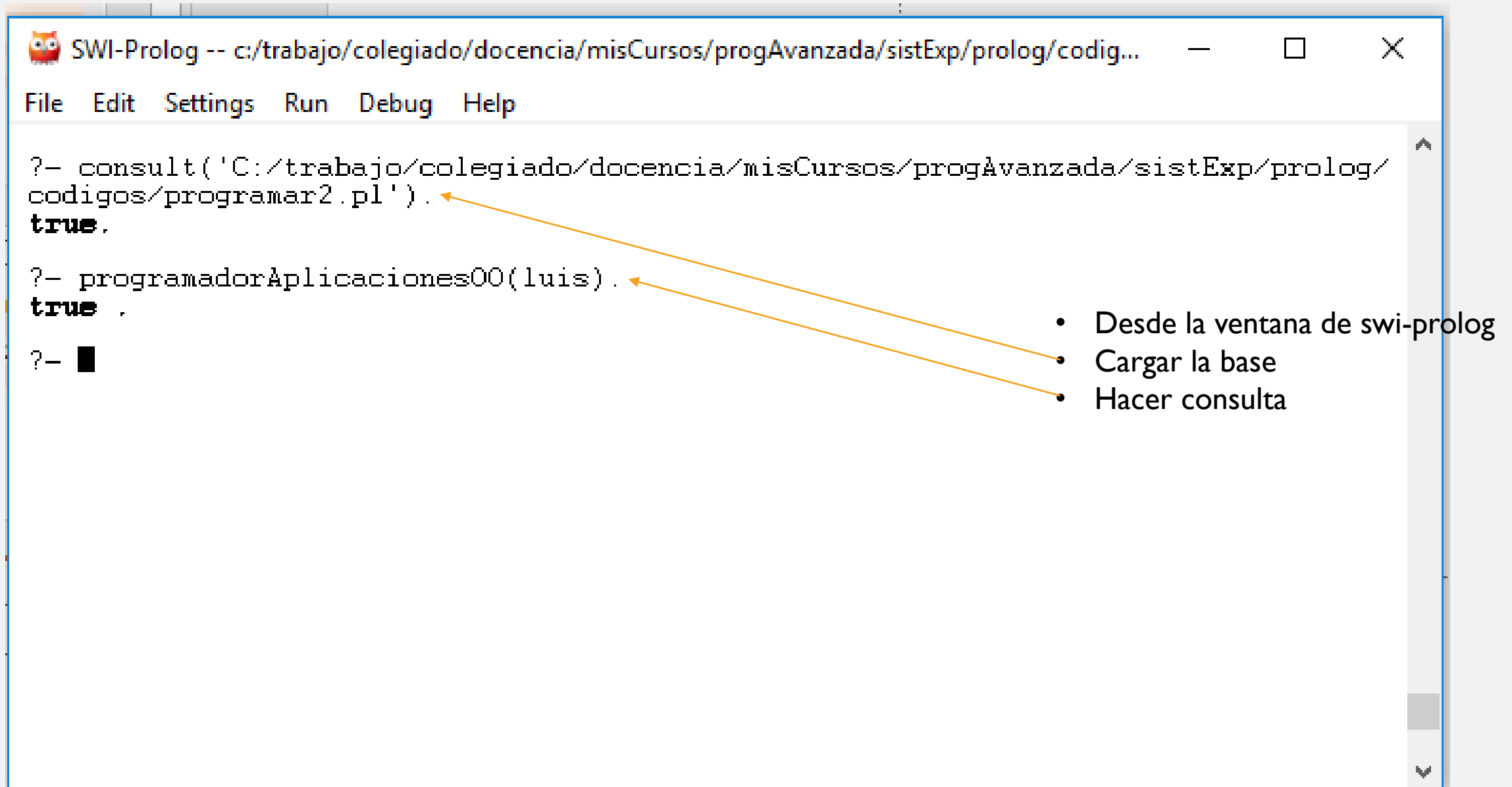


# GUARDE ASÍ: SELECCIONE



```
aprobo(ana,programacion1).
aprobo(luis,programacion1).
aprobo(luis,programacion2).
aprobo(luis,estructuraDatos).
aprobo(luis,poo).
reprobo(rosa,programacion1). /* rosa reprobo programacion1*/
reprobo(ana,programacion2).
/*X es programador básico si X aprobo programación 1*/
programadorBasico(X):-aprobo(X, programacion1).
/*X es programador de aplicaciones estucturadas si X aprobo programación 1 programación 2 y estructura de datos*/
programadorAplicacionesEstructuradas(X):-aprobo(X, programacion1),aprobo(X, programacion2),aprobo(X, estructuraDatos).
/*X es programador OO si X es programador de aplicaciones estructuradas y aprobo POO*/
programadorAplicacionesOO(X):-programadorAplicacionesEstructuradas(X),aprobo(X,poo).
```

# HACER CONSULTAS



```
SWI-Prolog -- c:/trabajo/colegiado/docencia/misCursos/progAvanzada/sistExp/prolog/codig...
File Edit Settings Run Debug Help

?- consult('C:/trabajo/colegiado/docencia/misCursos/progAvanzada/sistExp/prolog/
codigos/programar2.pl').
true.

?- programadorAplicaciones00(luis).
true.

?-
```

- Desde la ventana de swi-prolog
- Cargar la base
- Hacer consulta

## EJERCICIO

Suponga que queremos representar la regla:

La persona  $X$  es programador de propósito general si es programador de aplicaciones estructuradas o programador de aplicaciones OO.

¿Cómo escribiría esa regla en Swi-Prolog?

(Nota: recuerde que `;` representa el `or`)