Prolog

Comentarios:

/* Texto con comentarios… */
% Texto con comentarios…

Hechos: Empiezan con minúscula.

yes = cierto
true = cierto
false = falso
fail = predicado siempre falso

Variables: Empiezan con mayúscula.

Tipos: symbol, string, integer, real.
Listas: symbol *, integer *.

Operadores:
+ Aritméticos:

+	Suma
-	Resta
*	Multiplicación
/	División
mod	Módulo de la división

+ Relacionales:

>	Mayor que
>=	Mayor o igual que
<	Menor que
<=	Menor o igual que
<>	Distinto que
\=	Distinto que
=	Igual que
is	Igual que
=:=	Igualdad aritmética
=\=	Desigualdad aritmética

+ Lógicos:

,	Intersección
;	Conjunción
not(X)	Negación

+ El = funciona como asignador, para variables que no han sido inicializadas.

Los datos:

X = [x, y, ... , z]	Lista
X = ‘x y … z’	[‘x’, ‘y’, … , ‘z’]

X = [A | B]	A = Primer elemento B = Resto de la lista
_	Variable arbitraria. Da igual su contenido ya que no va a ser usada en la función.
nombre (var, …)	Registro

Predicados:

+ Consiste en las sentencias que pueden validar nuestras preguntas:

```
nombre (hecho, …) :- sentencias.
```

+ Si se pone “nombre (Variable, …)” da por pantalla, los hechos que pueden valer Variable, para que el predicado sea cierto.

+ El :- es el operador if, sirviendo para hacer cosas como:

```
pepe (X) :- X = jose.

pepe (juan)     = no
pepe (jose)     = yes

pepe (MiVar)    = jose
```

+ Para separar las sentencias se usan el operador conjunción (;) o el operador intersección (,). El operador not(X) es en realidad la siguiente función:

```
not (X) :- X, !, fail.
```

Funciones varias:

consult (rutafichero)	Carga un programa
reconsult (rutafichero)	Recarga un programa
fail	Da siempre falso (no)
write (dato)	Escribe en pantalla el dato
tell (rutaficherosalida)	Indica a write cual es el fichero de salida
nl	Produce un salto de línea
atom (X)	Indica si X es un átomo
var (X)	Indica si X es una variable
integer (X)	Indica si X es un entero

Estructura del programa:

```
Domains
    /* Declaración de los tipos de datos (listas, registros) */
    nombre = tipo.

Predicates
    /* Declaración de los prototipos de las funciones */
    nombre (tipoParametro, …).

Clauses
    /* Especificación de las funciones */
    nombre (Parametro, …) :- sentencias.
```

```
Goal
        /* Llamadas a las funciones */
        nombre (valor, …).
```

(Nota: El símbolo ! produce un break en la recursividad, cuando se buscan todas las alternativas que den yes en un predicado. Con !, si se ha dado ya un yes, se realiza el break.

Ejemplo de programa:

```
Domains
        nota = integer.

Predicates
        aprobado (nota).

Clauses
        aprobado (X) :- X = 5, !.
        aprobado (X) :- X = 6, !.
        aprobado (X) :- X = 7, !.
        aprobado (X) :- X = 8, !.
        aprobado (X) :- X = 9, !.
        aprobado (X) :- X = 10.

Goal
        aprobado (4).
```