7.1 Entrada y Salida

PDC Prolog incluye tres predicados de escritura. Estos son: **write**, **nl** y **writef**.

write: El predicado write tiene un número arbitrario de argumentos:

write(Arg1, Arg2, ..., ArgN).

Estos argumentos pueden ser constantes o pueden ser variables instanciadas.

nl: Mueve el cursor a la siguiente línea. Este predicado no tiene argumentos.

A continuación se explica cómo usar estos predicados para construirse uno sus propios predicados de escritura.

Ejemplo 7.1

```
domains
    lista = integer*

predicates
    writelista(lista)

clauses
    writelista([ ]).
    writelista([Ca | Co]):- write(Ca, " "), writelista(Co).
```

Ejercicio 7.1 Construye un programa que escriba los elementos de una lista con no más de cinco elementos por línea.

Ejemplo 7.2 Construir un programa que escriba una expresión de la forma

mas(mul(numero(99),x),mul(numero(3),x))

como

$99 * x + 3 * x$.

Ejemplo 7.2

domains

exp = numero(integer); mas(exp,exp); mul(exp,exp); x

predicates

escribir(exp)

clauses

```
escribir(x) :- write("x").
escribir(numero(N)) :- write(N).
escribir(mas(Exp1,Exp2)) :- escribir(Exp1),
    write(" + "),
    escribir(Exp2).
escribir(mul(Exp1,Exp2)) :- escribir(Exp1),
    write(" * "),
    escribir(Exp2).
```

7.2 Diseñar un programa: el cuadrado mágico.

Los cuadrados mágicos son cuadrados $n \times n$ de números, estos números son enteros desde 1 hasta $n * n$. Las filas, las columnas y las diagonales principales tienen que sumar el mismo número. Este número puede ser calculado sumando todos los enteros de 1 a $n \times n$ y dividiendo el resultado por el número de filas.

Como ejemplo, consideramos un cuadrado mágico 3×3. Este debería ser llenado con enteros desde 1 hasta $3 * 3 = 9$. Las filas, columnas y diagonales principales deben de sumar 15.

7.2.1 Entrada y salida: No hay entradas y queremos una salida, por ejemplo, como ésta:

6	1	8
7	5	3
2	9	4

7.2.2 Estructura de los datos: Hay dos estructuras importantes de datos. La primera contendrá los números del 1 al 9 y lo representaremos por la lista [1,2,3,4,5,6,7,8,9].

La segunda representará el cuadrado. Se puede codificar como una lista representando la primera celda como a, la segunda como b, etc.

a	b	c
d	e	f
g	h	i

Nos ayudaría conocer de alguna forma qué celda se está procesando en el programa. De aquí, que sea conveniente codificar cada celda como una estructura de dos argumentos, el primero será la letra que identifica la celda y el segundo argumento contendrá el entero.

Por tanto, el cuadrado será representado por la lista

[celda(a,N1), celda(b, N2), celda(3, N3), ... celda(i, Ni)].

7.2.3 El programa: Hay tres procesos básicos:

- Generar un cuadrado.
- Comprobar que el cuadrado es mágico.
- Dibujar el cuadrado.

domains

celdas = celda(symbol, integer)

lista = integer*

cuadrados = celdas*

predicates

cuadrado(cuadrados)

numeros(lista)

suma(integer)

llenar_cuadrado(lista,cuadrados)

comprobar_cuadrado(cuadrados)

generar_cuadrado

pertenece(celdas,cuadrados)

dibujar(cuadrados)

clauses

cuadrado([celda(a,_),celda(b,_), . . . , celda(i,_)]).

numeros([1,2,3,4,5,6,7,8,9]).

suma(15).

generar_cuadrado:-

cuadrado(Cuadrado),
numeros(Numero),
llenar_cuadrado(Numero, Cuadrado),
comprobar_cuadrado(Cuadrado),
dibujar(Cuadrado).