UNIVERSDAD DE CÓRDOBA

ESCUELA POLITÉCNICA SUPERIOR

DEPARTAMENTO DE
INFORMÁTICA Y ANÁLISIS NUMÉRICO

LENGUAJES

DE INTELIGENCIA ARTIFICIAL

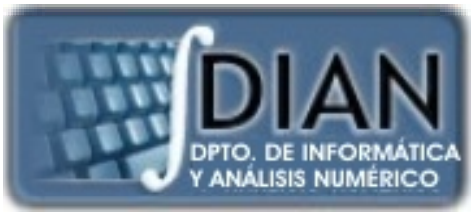
INGENIGERÍA TÉCNICA EN INFORMÁTICA DE GESTIÓN

INGENIERÍA TÉCNICA EN INFORMÁTICA DE SISTEMAS

SEGUNDO CURSO

PRIMER CUATRIMESTRE

CURSO ACADÉMICO 2009 - 2010



Primera parte: Scheme

- **Tema 1.-** Introducción al Lenguaje Scheme
- **Tema 2.-** Expresiones y Funciones
- **Tema 3.-** Predicados y sentencias condicionales
- **Tema 4.-** Iteración y Recursión
- **Tema 5.-** Tipos de Datos Compuestos
- **Tema 6.-** Abstracción de Datos
- **Tema 7.-** Lectura y Escritura

Segunda parte: Prolog

- **Tema 8.-** Introducción al Lenguaje Prolog
- **Tema 9.-** Elementos Básicos de Prolog
- **Tema 10.-** Listas
- **Tema 11.-** Reevaluación y el "corte"
- **Tema 12.-** Entrada y Salida

Segunda parte: Prolog

Tema 8.- Introducción al Lenguaje Prolog

Tema 9.- Elementos Básicos de Prolog

Tema 10.- Listas

Tema 11.- Reevaluación y el "corte"

Tema 12.- Entrada y Salida

1. Escritura

- **write** y **display**: escriben
  - Números
  - Átomos
  - Estructuras
  - Variables
  - Cadenas: códigos ASCII

- **put**: escribe un carácter

- **nl**: escribe un salto de línea

- **tab**: escribe espacios en blanco

1. Escritura

- write(número).

  ? write(12).
  12

- write(átomo).

  ?write(agua).
  agua

  ?write(‘Córdoba’).
  Córdoba

- write(estructura).

  ? write (autor(juan, valera)).
  autor(juan,valera).

  ?write(3+2)
  3+2

- write(lista).

  ? write ([a, b, c])).
  [a, b, c]

1. Escritura

- write("cadena"): escribe los códigos ASCII de sus caracteres

```
? write("hola").
 [72, 111, 109, 97]
```

- write(Variable):
  - Si la variable posee un valor entonces escribe su contenido

```
? factorial(3,N),  write(N).
 6
```

  - Si la variable no posee un valor entonces escribe su dirección de memoria

```
?  write(N).
 _G624
```

1. Escritura de términos y caracteres

- display(número).

  ? display(12).
  12

- display(átomo).

  ?display(agua).
  agua

  ?display(‘Córdoba’).
  Córdoba

- display(estructura).

  ? display (autor(juan, valera)).
  autor(juan,valera).

  ?display(3+2).
  +(3, 2)

- display(lista).

  ? display ([a, b, c])).
  .(a, .(b, .(c, [])))

1. Escritura

- display(“cadena”): escribe los códigos ASCII de sus caracteres

```
? display(“hola”).
 .(72, .(111, .(108, .(97, []))))
```

- display(Variable):
  - Si la variable posee un valor entonces escribe su contenido

```
? factorial(3,N),  display(N).
 6
```

  - Si la variable no posee un valor entonces escribe su dirección de memoria

```
?  display(N).
 _G624
```

1. Escritura

•put(N): escribe el carácter cuyo código ASCII es N.

```
?put(104), put(111), put(108), put(97)
 hola
```

• tab(N): escribe N espacios en blanco

```
? tab(3).
   _ _ _
```

• tab se puede definir como

```
tab(0):-!.
tab(N):- put(32), N1 is N-1, tab(N1).
```

1. Escritura

```
escribir_cadena([]).

escribir_cadena([Cabeza|Cola]):-
        put(Cabeza),
        escribir_cadena(Cola).
```

? escribir_cadena(“Cadena maravillosa”).
Cadena maravillosa

/* Escritura de los elementos de una lista */

```
/* Escribe una lista en una fila */

escribir_fila([]).

escribir_fila([X|Y]):-
                write(X),
                tab(1),
                escribir_fila(Y).

escribir_fila(X):-
                write(X),
                tab(1).
```

```
?- escribir_fila([1,2,3,4]).
1 2 3 4
```

/* Escritura de los elementos de una lista */

/* Escribe una lista en una columna */

```
escribir_columna([]).

escribir_columna([X|Y]):-
                write(X),
                nl,
                escribir_columna(Y).

escribir_columna([X|Y]):-
                escribir_columna(X),
                nl,
                escribir_columna(Y).
```

```
?- escribir_fila([1,2,3,4]).
 1
 2
 3
 4
```

```
/*  Escribe una lista con sublistas de forma sangrada
 *  a partir de la columna indicada.
 *
 * Las sublistas tienen una sangría de 3 espacios.
 *
 * Se usa la negación con not
 */

/* El argumento no es una lista */
escribir_lista(X,Columna):- not(es_lista(X)),
                            tab(Columna),
                            write(X),
                             nl.

/* El argumento es una lista con cabeza y cola */
escribir_lista([Cabeza|Cola],Columna):-
                          Lugar is Columna + 3,
                          escribir_lista(Cabeza,Lugar),
                          escribir_sublista(Cola,Lugar),
                          nl.
```

```
/* Si la sublista es vacía, no escribe nada */
escribir_sublista([],_).

/* Si la sublista no es vacía, se escribe la cabeza y la cola */

escribir_sublista([Cabeza|Cola],Columna):-
                                escribir_lista(Cabeza,Columna),
                                escribir_sublista(Cola,Columna).


/* Se comprueba si el argumento es una lista */

/* Es la lista vacía */
es_lista([]).

/* Es una lista que posee cabeza y cola */
es_lista([_|_]).
```

?- escribir lista([1,2,[3,4],5,6,[7,[8,9],10],2).

1

2

3

4

5

6

7

8

9

10

```
/*  Escribe una lista con sublistas de forma sangrada
 *  a partir de la columna indicada.
 *
 * Las sublistas tienen una sangría de 3 espacios.
 *
 * Se usa el corte
 */

/* El argumento es una Lista que posee Cabeza y Cola */
escribir_lista([Cabeza|Cola],Columna):-
                              !,
                              Lugar is Columna + 3,
                              escribir_lista(Cabeza,Lugar),
                              escribir_sublista(Cola,Lugar),
                              nl.

/* El argumento es un elemento */
escribir_lista(X,Columna):-
                              tab(Columna),
                              write(X),
                              nl.
```

```
/* Si es la sublista está vacía, no escribe nada */
escribir_sublista([],_).


/* El argumento es una Sublista con Cabeza y Cola */

escribir_sublista([Cabeza|Cola],Columna):-
                                escribir_lista(Cabeza,Columna),
                                escribir_sublista(Cola,Columna).
```

```
/* Problema de las torres de Hanoi */

hanoi(N):- mover(N,izquierda,centro,derecha).

mover(1,A,_,C):- escribir_movimiento(A,C), !.

mover(N,A,B,C):- N1 is N-1,
                 mover(N1,A,C,B),
                 escribir_movimiento(A,C),
                 mover(N1,B,A,C).

escribir_movimiento(Origen, Destino):-
                 nl,
                 write(Origen),
                 write(' --> '),
                 write(Destino).
```

?- hanoi(3).

izquierda --> derecha
izquierda --> centro
derecha --> centro
izquierda --> derecha
centro --> izquierda
centro --> derecha
izquierda --> derecha

Yes

2. Lectura

- **read(X)**
  - Lee el siguiente **término**, que deber finalizar con punto “.”

- **get0(X)**
  - Lee el siguiente **carácter** que se teclee.

- **get(X)**
  - Lee el siguiente **carácter** que se teclee pero **desprecia** los caracteres **no imprimibles**.

2. Lectura

- read(X).
  - Lee el siguiente término, que deber finalizar con punto “.”

?read(X).
| agua.
X = agua

- Si X posee un valor antes de la lectura entonces se comprueba si coincide con el valor leído.

? factorial(3,N), read(N).
| 6.
N = 6

2. Lectura

- get0(X).

```
?get0(X).
|a
X = 97
?get0(X).
| Enter
X = 10
```

- get(X).

```
?get(X).
|a
X = 97
?get(X).
| Enter
| Enter
| b
X = 98
```

2. Lectura

```
/* padre_de(P1,P2): P2 es el padre de P1 */
padre_de(juan, miguel).
padre_de(marta,miguel).
padre_de(carmen,miguel).

mostrar_padre:- write('Nombre --> '),
                read(X),
                write('El padre de '),
                write(X),
                write(' es '),
                padre_de(X,Y),
                write(Y).

? mostrar_padre.
Nombre --> juan.
El padre de juan es miguel
Yes
```

```
/* Lectura de una frase y transformación en átomos */

leer_frase(Palabras):-
                    get0(Caracter),
                    leer_resto(Caracter,Palabras).

/* El punto "." indica el fin de la frase */
leer_resto(46,[]):- !.

/* Se omite el espacio en blanco */
leer_resto(32,Palabras):-
                    !,
                    leer_frase(Palabras).


/* Lee los caracteres de la palabra actual */

leer_resto(Caracter,[Palabra|Palabras]):-
                    leer_caracteres(Caracter,Caracteres,Siguiente_caracter),
                    name(Palabra,Caracteres),
                    leer_resto(Siguiente_caracter,Palabras).
```

```
/* Fin de palabra: 46 = punto "." */
leer_caracteres(46,[],46):- !.

/* Fin de palabra: 32 = espacio en blanco */
leer_caracteres(32,[],32):- !.


leer_caracteres(Caracter,[Caracter|Caracteres],Siguiente_caracter):-
        get0(Nuevo_Caracter),
        leer_caracteres(Nuevo_Caracter,Caracteres,Siguiente_caracter).
```

```
?- leer_frase(X).
| Esta frase    va  a ser    transformada.

X = [‘Esta’, frase, va, a, ser, transformada]
```

3. Lectura y escritura con un fichero

- Los nombres de los ficheros se representan con átomos:
  ‘ C:\datos.txt’

- Fichero predeterminado de lectura
  User

- Fichero predeterminado de escritura
  La pantalla

3. Lectura y escritura con un fichero

- see(argumento)
  - Apertura de un fichero para leer
  - El argumento pasa a ser el fichero de lectura actual de read, get0 y get.

- see(X)
  abre el fichero que indique X

- see('entrada.txt')
  abre el fichero entrada.txt

- see('C:\entrada.txt')
  abre el fichero C:\entrada.txt

3. Lectura y escritura con un fichero

- seeing: comprueba el fichero que está abierto para lectura
  - seeing('datos').
    Es cierto si datos está abierto para lectura

  - seeing(X).
    - Si X no tiene un valor entonces le asigna a X el valor del fichero de lectura actual
    - Si X tiene un valor, se comprueba si coincide con el valor del fichero de lectura actual.

- seen: cierra el fichero de lectura actual.

3. Lectura y escritura con un fichero

```
/* Se numeran los elementos leídos a partir de N */
contar(N):-
    read(Termino),
    mostrar(Termino,N).

mostrar(end_of_file,_):- !.

mostrar(Termino,N):-
    write(N),
    tab(2),
    write(Termino),
    nl,
    N1 is N + 1,
    contar(N1).
```

3. Lectura y escritura con un fichero

```
/* Contenido del fichero entrada.txt */
agua.
fuego.
tierra.
aire.
/* Fin del contenido del fichero */

? see('entrada.txt'), contar(1).
?- contar(1).
1  agua
2  fuego
3  tierra
4  aire

Yes
```

3. Lectura y escritura con un fichero

- tell(argumento)
  - Apertura de un fichero para escribir
  - El argumento pasa a ser el fichero de escritura actual de write, display, tab, nl y put.
  - tell(X)
    abre el fichero que indique X

  - tell('salida.txt')
    abre el fichero salida.txt

  - tell('C:\salida.txt')
    abre el fichero C:\salida.txt

3. Lectura y escritura con un fichero

- telling: comprueba el fichero que está abierto para escritura
  - telling('datos').
    Es cierto si datos está abierto para escritura
  - telling(X).
    - Si X no tiene un valor entonces le asigna a X el valor del fichero de escritura actual
    - Si X tiene un valor, se comprueba si coincide con el valor del fichero de escritura actual.
- told: cierra el fichero de escritura actual.

3. Lectura y escritura con un fichero

```
/* Contenido del fichero entrada.txt */
agua.
fuego.
tierra.
aire.
/* Fin del contenido del fichero */

?see('entrada.txt'),tell('salida.txt'),contar(1),told,seen.
Yes

/* Contenido del fichero salida.txt */
1  agua
2  fuego
3  tierra
4  aire
```

4. Consulta de ficheros

- ['programa'].
  - Carga programa.pl
- consult('programa').
  - Carga programa.pl
- ['programa1', 'programa2'].
  - Carga programa1.pl y programa2.pl
- reconsult('programa').
  - Carga programa.pl y sustituye las reglas redefinidas.

UNIVERSDAD DE CÓRDOBA

ESCUELA POLITÉCNICA SUPERIOR

DEPARTAMENTO DE
INFORMÁTICA Y ANÁLISIS NUMÉRICO

LENGUAJES

DE INTELIGENCIA ARTIFICIAL

INGENIGERÍA TÉCNICA EN INFORMÁTICA DE GESTIÓN

INGENIERÍA TÉCNICA EN INFORMÁTICA DE SISTEMAS

SEGUNDO CURSO

PRIMER CUATRIMESTRE

CURSO ACADÉMICO 2009 - 2010