



UNIVERSIDAD DE CÓRDOBA

ESCUELA POLITÉCNICA SUPERIOR DE CÓRDOBA

DEPARTAMENTO DE  
INFORMÁTICA Y ANÁLISIS NUMÉRICO



# PROGRAMACIÓN DECLARATIVA

INGENIERÍA INFORMÁTICA

CUARTO CURSO

PRIMER CUATRIMESTRE

**Tema 12.- Entrada y salida**



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**

# Índice

1. Apertura de ficheros y cierre de flujos
2. Lectura y escritura de términos
3. Lectura y escritura de caracteres
4. Modificación de los dispositivos de entrada y salida actuales

# Índice

1. **Apertura de ficheros y cierre de flujos**
2. Lectura y escritura de términos
3. Lectura y escritura de caracteres
4. Modificación de los dispositivos de entrada y salida actuales

# 1. Apertura de ficheros y cierre de flujos

- Apertura de ficheros
- Cierre de flujos

# 1. Apertura de ficheros y cierre de flujos

- **Observación**

- **Nombres de los ficheros**

- Se representan como **átomos** de Prolog, escribiéndolos entre **comillas simples**.

- **Ejemplos**

*‘/home/usuario/fichero.txt’*

*‘salida.txt’*

- Fichero de entrada por defecto

- El **teclado** y se denomina: ***user***

- Fichero de salida por defecto

- La **pantalla** y se denomina: ***user***

# 1. Apertura de ficheros y cierre de flujos

- Apertura de ficheros
- Cierre de flujos



# 1. Apertura de ficheros y cierre de flujos

- Apertura de ficheros

- *open*

# 1. Apertura de ficheros y cierre de flujos

- Apertura de ficheros

- *open*

- Sintaxis

*open*(Argumento, Modo, Variable)

- Argumento:

- ✓ Designa el fichero que se desea abrir

- Modo:

- ✓ *read*: lectura

- ✓ *write*: escritura

- Variable

- ✓ Representa el flujo asociado al fichero<sup>10</sup>

# 1. Apertura de ficheros y cierre de flujos

- Apertura de ficheros

- *open*

- Descripción

- read*

- ✓ Abre el fichero para lectura

- ✓ El fichero **debe** existir

- write*

- ✓ Abre el fichero para escritura

- ✓ El fichero es **creado**

# 1. Apertura de ficheros y cierre de flujos

- Apertura de ficheros

- *open*

- Ejemplos

*open*('entrada.txt', *read*, X)

*Nombre* = 'entrada.txt', *open*(*Nombre*, *read*, X).

*open*('salida.txt', *write*, Y)

*Nombre* = 'salida.txt', *open*(*Nombre*, *write*, Y).

# 1. Apertura de ficheros y cierre de flujos

- Apertura de ficheros
- **Cierre de flujos**

# 1. Apertura de ficheros y cierre de flujos

- Cierre de flujos

- *close*

# 1. Apertura de ficheros y cierre de flujos

- Cierre de flujos

- *close*

- Sintaxis

*close*(Argumento)

- Argumento

- ✓ Término asociado a un flujo

- Descripción

- Cierra el flujo indicado por el argumento.

- Si el flujo **no existe**, se genera un **error**.

# 1. Apertura de ficheros y cierre de flujos

- Cierre de flujos

- *close*

- Ejemplos

- ?- *open*('entrada.txt', read, X), ..., *close* (X).

- ?- *close*(X).

- Error*



# Índice

1. Apertura de ficheros y cierre de flujos
- 2. Lectura y escritura de términos**
3. Lectura y escritura de caracteres
4. Modificación de los dispositivos de entrada y salida actuales

## 2. Lectura y escritura de términos

- Escritura
- Lectura
- Ejemplo de lectura y escritura de fichero con átomos
- Ejemplo de carga de los átomos de un fichero en una lista

## 2. Lectura y escritura de términos

- **Escritura**
- Lectura
- Ejemplo de lectura y escritura de fichero con átomos
- Ejemplo de carga de los átomos de un fichero en una lista

## 2. Lectura y escritura de términos

- Escritura
  - *write* y *display*

## 2. Lectura y escritura de términos

- **Escritura**

- ***write* y *display***

- **Sintaxis**

*write*([Flujo,] Argumento)

*display*([Flujo,] Argumento)

- **Flujo :**

- ✓ Flujo asociado a un fichero abierto para escritura.

- **Argumento:**

- ✓ número, átomo, estructura, lista, etc.

## 2. Lectura y escritura de términos

- **Escritura**

- ***write* y *display***

- **Descripción**

- Escribe el argumento en el dispositivo de salida actual (*current\_output*)

*write*(Argumento)

*display*(Argumento)

- Escribe el argumento en el flujo indicado

*write*(Flujo, Argumento)

*display*(Flujo, Argumento)

- **Observación**

- Se escribirá en el fichero al cerrar el flujo.

## 2. Lectura y escritura de términos

- **Escritura**

- ***write* y *display***

- **Observación**

- ***display***

- ✓ muestra la **representación interna** de las estructuras,
- ✓ considerando como tales a las listas y las expresiones aritméticas.

## 2. Lectura y escritura de términos

- **Escritura**

- *write* y *display*

- **Ejemplos**

- **Números, átomos y estructuras.**

<i>?-write(12).</i> 12	<i>?-display(12).</i> 12
<i>?-write(luz).</i> luz	<i>?-display(luz).</i> luz
<i>?-write(autor('Juan', 'Varela')).</i> <i>autor(Juan, Varela).</i>	<i>?-display(autor('Juan', 'Varela')).</i> <i>autor(Juan, Varela).</i>



## 2. Lectura y escritura de términos

- **Escritura**

- ***write* y *display***

- **Ejemplos**

- **Variables (1/3)**

- ✓ Si una variable **no está instanciada**, muestra **su dirección de memoria**

?- ***write(X), display(X).***

***\_G2062\_G2062***

***true.***

## 2. Lectura y escritura de términos

- **Escritura**

- ***write* y *display***

- **Ejemplos**

- **Variables (2/3)**

- ✓ Si una variable está **instanciada**, muestra su **valor**

?- *factorial(3,R), write(R),tab(1), display(R).*

6 6

*R = 6*

*true.*

## 2. Lectura y escritura de términos

- **Escritura**

- *write* y *display*

- **Ejemplos**

- **Variables (3/3)**

- ✓ Si dos variables “**comparten**” memoria y no están instanciadas, se muestra la **misma dirección de memoria**.

?- *X=Y, write(X),display(X), write(Y),display(Y).*

*\_G2868\_G2868\_G2868\_G2868*

*X = Y*

## 2. Lectura y escritura de términos

- **Escritura**

- ***write* y *display***

- **Ejemplos**

- **Expresiones aritméticas**

- ✓ ***write*** **no** evalúa la expresión, pero muestra cada uno de los argumentos.
- ✓ ***display*** **no** evalúa la expresión, pero muestra su **representación como estructura**.

<b>?-write(2+3).</b> 2+3	<b>?-display(2+3).</b> +(2,3)
<b>?-X is 2, write(X+3).</b> 2+3	<b>?-X is 2, display(X+3).</b> +(2,3)

## 2. Lectura y escritura de términos

- **Escritura**

- ***write* y *display***

- **Ejemplos**

- **Listas**

- ✓ ***write*** muestra cada uno de los argumentos.

- ✓ ***display*** muestra la **representación interna**.

<pre>?-write([1,2,3]). [1,2,3].</pre>	<pre>?- display([1,2,3]). .(1,.(2,.(3,[])))</pre>
<pre>?- X is 2, write([1,X,Y]). [1,2,_G326] X = 2</pre>	<pre>?- X is 2, display([1,X,Y]). .(1,.(2,.(<u>_G338</u>,[]))) X = 2</pre>

## 2. Lectura y escritura de términos

- **Escritura**

- ***write* y *display***

**Observación:**  
**antiguas versiones de Prolog**

- **Ejemplos**

- **Cadenas de caracteres**

- ✓ ***write*** muestra una lista de **códigos ASCII**
- ✓ ***display*** muestra la **representación interna** de dicha lista.

<pre>?- write("Hola"). [72,111,108,97] true.</pre>	<pre>?- display("Hola"). .(72,.(111,.(108,.(97,[]))) true.</pre>
--	--

## 2. Lectura y escritura de términos

- **Escritura**

- ***write* y *display***

- **Ejemplos**

- **Escritura en un fichero**

***open***('nuevo.txt', ***write***,***X***),

*A is 2\*3,*

***write***(***X***,***A***),

***close***(***X***).

## 2. Lectura y escritura de términos

- **Escritura**

- **Ejemplos**

- Torres de Hanoi
    - Escritura de listas



## 2. Lectura y escritura de términos

- **Escritura**

- **Ejemplos**

- **Torres de Hanoi**
- **Escritura de listas**

## 2. Lectura y escritura de términos

- **Escritura**

- **Ejemplos**

- **Torres de Hanoi**

- **Primera parte**

*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).*

## 2. Lectura y escritura de términos

- **Escritura**

- **Ejemplos**

- **Torres de Hanoi**

- Segunda parte

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

## 2. Lectura y escritura de términos

- **Escritura**

- **Ejemplos**

- **Torres de Hanoi**

?- *hanoi(3).*

*izquierda --> derecha*

*izquierda --> centro*

*derecha --> centro*

*izquierda --> derecha*

*centro --> izquierda*

*centro --> derecha*

*izquierda --> derecha*

*true.*

## 2. Lectura y escritura de términos

- **Escritura**

- **Ejemplos**

- Torres de Hanoi
- **Escritura de listas**

## 2. Lectura y escritura de términos

- **Escritura**

- **Ejemplos**

- **Escritura de listas**

- Escritura de una lista en una fila

- Escritura de una lista en una columna

- Escritura sangrada de una lista con sublistas

## 2. Lectura y escritura de términos

- **Escritura**

- **Ejemplos**

- **Escritura de listas**

- Escritura de una lista en una fila *escribir\_fila([])*.

```
escribir_fila([Cabeza | Cola]):-  
    write(Cabeza),  
    tab(1),  
    escribir_fila(Cola).
```

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

## 2. Lectura y escritura de términos

- **Escritura**

- **Ejemplos**

- **Escritura de listas**

- Escritura de una lista en una **columna**  
*escribir\_columna([]).*

```
escribir_columna([Cabeza | Cola]):-  
    write(Cabeza),  
    nl,  
    escribir_columna(Cola).
```



## 2. Lectura y escritura de términos

- **Escritura**

- **Ejemplos**

- **Escritura de listas**

- Escritura de una lista en una **columna**

?- *escribir\_columna([a,b,c,d]).*

*a*

*b*

*c*

*d*

*true*

## 2. Lectura y escritura de términos

- **Escritura**

- **Ejemplos**

- **Escritura de listas**

- **Escritura sangrada de una lista con sublistas**

- ✓ **Versión con “not” (1/6)**

*/\* El argumento no es una lista \*/*

*escribir\_lista(X,Columna):-*

*not(es\_lista(X)),*

*tab(Columna),*

*write(X),*

*nl.*

## 2. Lectura y escritura de términos

- **Escritura**

- **Ejemplos**

- **Escritura de listas**

- Escritura sangrada de una lista con sublistas

- ✓ Versión con “not” (2/6)

*/\* 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).*

## 2. Lectura y escritura de términos

- **Escritura**

- **Ejemplos**

- **Escritura de listas**

- **Escritura sangrada de una lista con sublistas**

- ✓ Versión con “not” (3/6)

*/\* Si la sublista es vacía, no escribe nada \*/*

*escribir\_sublista([],\_).*

## 2. Lectura y escritura de términos

- **Escritura**

- **Ejemplos**

- **Escritura de listas**

- Escritura sangrada de una lista con sublistas

- ✓ Versión con “not” (4/6)

*/\* 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).*

## 2. Lectura y escritura de términos

- **Escritura**

- **Ejemplos**

- **Escritura de listas**

- **Escritura sangrada de una lista con sublistas**

- ✓ Versión con “**not**” (5/6)

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

*/\* Es la lista vacía \*/*

`es_lista([]).`

*/\* Es una lista que posee cabeza y cola \*/*

`es_lista([_ | Cola):- es_lista(Cola).`

## 2. Lectura y escritura de términos

- **Escritura**

- **Ejemplos**

- **Escritura de listas**

- **Escritura sangrada de una lista con sublistas**

- ✓ Versión con “**not**” (6/6)

- ?- *escribir\_lista([a,[b,c],d,[e]],10).*

- a*

- b*

- c*

- d*

- e*

- true.*

## 2. Lectura y escritura de términos

- **Escritura**

- **Ejemplos**

- **Escritura de listas**

- Escritura sangrada de una lista con sublistas

- ✓ Versión con “**el corte**” (1/4)

*/\* 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).*



## 2. Lectura y escritura de términos

- **Escritura**

- **Ejemplos**

- **Escritura de listas**

- Escritura sangrada de una lista con sublistas

- ✓ Versión con “**el corte**” (2/4)

*/\* El argumento es un elemento \*/*

*escribir\_lista(X,Columna):-*

*tab(Columna),*

*write(X),*

*nl.*

## 2. Lectura y escritura de términos

- **Escritura**

- **Ejemplos**

- **Escritura de listas**

- Escritura sangrada de una lista con sublistas

- ✓ Versión con “**el corte**” (3/4)

*/\* 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)<sup>50</sup>.*

## 2. Lectura y escritura de términos

- **Escritura**

- **Ejemplos**

- **Escritura de listas**

- **Escritura sangrada de una lista con sublistas**

- ✓ Versión con “**el corte**” (4/4)

- ?- *escribir\_lista([a,[b,c],d,[e]],10).*

- a*

- b*

- c*

- d*

- e*

- true.*

## 2. Lectura y escritura de términos

- Escritura
- **Lectura**
- Ejemplo de lectura y escritura de fichero con átomos
- Ejemplo de carga de los átomos de un fichero en una lista

## 2. Lectura y escritura de términos

- Lectura
  - *read*

## 2. Lectura y escritura de términos

- Lectura

- *read*

- Sintaxis

- read([Flujo,]Variable)*

## 2. Lectura y escritura de términos

- Lectura

- *read*

- Descripción

- ❑ Lee el siguiente término,
  - ✓ que debe terminar en punto “.”,
  - ✓ que esté disponible en *Flujo* o en el dispositivo de entrada actual (*current input device*)
  - ✓ que, por defecto, es el teclado.
- ❑ La variable quedará *instanciada* con el valor leído.
- ❑ Si la variable estuviera instanciada antes de la lectura, se *comprobará* si el término leído es igual al valor de la variable.

## 2. Lectura y escritura de términos

- Lectura

- *read*

- Ejemplos

?- *read(X), write(X).*

|: 1.

1

*X = 1.*

?- *read(X), write(X).*

|: *agua.*

*agua*

*X = agua.*

Se escribe el punto “.”  
para finalizar



## 2. Lectura y escritura de términos

- Lectura

- *read*

- Ejemplos

?- *read(X), write(X).*

|: 2+3.

2+3

*X = 2+3.*

?- *read(X), write(X).*

|: *autor('Juan','Varela').*

*autor(Juan,Varela)*

*X = autor('Juan', 'Varela').*

## 2. Lectura y escritura de términos

- Lectura

- *read*

- Ejemplos

?- *X is 2, read(X).*

|: 2.

*X = 2.*

?- *X is 2, read(X).*

|: 3.

*false*

## 2. Lectura y escritura de términos

- Lectura

- *read*

- Ejemplos

?- *read(X), write(X).*

|: *Dato.* ←

El **Dato** leído es una variable

*\_G287*

*true.*

## 2. Lectura y escritura de términos

- Lectura

- *read*

- Ejemplos

?- *read(X), write(X).*

|: *[a,b,c].*

*[a,b,c]*

*X = [a, b, c].*

?- *read(X), write(X).*

|: *[a,B,c].*

*[a,\_G411,c]*

*X = [a, \_G411, c].*

## 2. Lectura y escritura de términos

- Lectura

- *read*

- Ejemplos

*padre(juan,miguel).*

*padre(marta,miguel).*

*padre(carmen,miguel).*

*buscar\_padre:- write('Nombre --> '),*

***read**(X),*

*write('Padre de '), write(X), write(' es '),*

*padre(X,Y),*

*write(Y).*

## 2. Lectura y escritura de términos

- Lectura

- *read*

- Ejemplos

?- *buscar\_padre.*

*Nombre --> marta.*

*Padre de marta es miguel*

*true.*

## 2. Lectura y escritura de términos

- Lectura

- *read*

- Ejemplos

*/\* Contenido del fichero entrada.txt \*/*

*agua.*

*fuego.*

*tierra.*

*aire.*

?- *open('entrada.txt',read,X), read(X,A), read(X,B), close(X).*

*X = <stream>(0x13a7ef0),*

*A = agua,*

*B = fuego.*

## 2. Lectura y escritura de términos

- Escritura
- Lectura
- **Ejemplo de lectura y escritura de fichero con átomos**
- Ejemplo de carga de los átomos de un fichero en una lista



## 2. Lectura y escritura de términos

- Ejemplo de lectura y escritura de fichero con átomos

- `copy_atoms_file.pl`

- copia de un fichero con átomos (1/3)

`copy(InputFile,OutputFile) :-`

`open(InputFile,read,InputStream), % Open for reading`

`open(OutputFile,write,OutputStream), % Open for writing`

`repeat, % Loop driving by failure`

`read(InputStream, Atom), % read the atom`

`process(OutputStream,Atom), % process the atom`

`% Close the streams`

`close(InputStream),`

`close(OutputStream),`

`!. % Stop now.`

## 2. Lectura y escritura de términos

- Ejemplo de lectura y escritura de fichero con átomos

- `copy_atoms_file.pl`

- copia de un fichero con átomos (2/3)

*% Atom of end of file*

*process*(\_,end\_of\_file) :- !.

*% Process the atom*

*process*(OutputStream,Atom) :-

*write*(OutputStream,Atom), *% Write the atom*

*put*(OutputStream,46), *% Write the dot*

*nl*(OutputStream), *% New line*

*% Loop driving by failure*

*fail.*

## 2. Lectura y escritura de términos

- Ejemplo de lectura y escritura de fichero con átomos
  - `copy_atoms_file.pl`
    - copia de un fichero con átomos (3/3)  
?- `copy` (*'entrada.txt'*, *'salida.txt'*).

## 2. Lectura y escritura de términos

- Escritura
- Lectura
- Ejemplo de lectura y escritura de fichero con átomos
- **Ejemplo de carga de los átomos de un fichero en una lista**

## 2. Lectura y escritura de términos

- Ejemplo de carga de los átomos de un fichero en una lista

- **atomos.txt**

*agua.*

1.

*fuego.*

2.

*tierra.*

3.

*aire.*

4.

?- **cargar**('atomos.txt',L).

*L = [agua, 1, fuego, 2, tierra, 3, aire, 4].*

## 2. Lectura y escritura de términos

- Ejemplo de carga de los átomos de un fichero en una lista

- cargarFicheroEnlista1.pl

*% MÉTODO 1*

*% Carga los átomos de un fichero en una lista*

*cargar*(FicheroEntrada,Lista) :-

*open*(FicheroEntrada,read,FlujoEntrada),

*leerFichero*(FlujoEntrada,Lista),

*close*(FlujoEntrada),

*!.*

*leerFichero*(FlujoEntrada,Resultado):-

*read*(FlujoEntrada,N),

(

*N \= end\_of\_file -> leerFichero*(FlujoEntrada,Lista),  
*Resultado = [N\Lista];*

*true -> Resultado = []*

*).*

## 2. Lectura y escritura de términos

- Ejemplo de carga de los átomos de un fichero en una lista

- cargarFicheroEnlista2.pl

*% MÉTODO 2*

*% Carga los átomos de un fichero en una lista*

*cargar*(FicheroEntrada,Lista) :-

*open*(FicheroEntrada,read,FlujoEntrada),

*leerFichero*(FlujoEntrada,Lista),

*close*(FlujoEntrada),

*!*.

*leerFichero*(FlujoEntrada,Lista):-

*read*(FlujoEntrada,N),

*procesar*(FlujoEntrada,N,Lista).

*procesar*(FlujoEntrada,N,[N|Cola]):-

*N \= end\_of\_file , !,*

*leerFichero*(FlujoEntrada,Cola).

*procesar*(\_,\_,[]).

## 2. Lectura y escritura de términos

- Ejemplo de carga de los átomos de un fichero en una lista

- cargarFicheroEnlista3.pl

*% MÉTODO 3*

*% Carga los átomos de un fichero en una lista*

```
cargar(FicheroEntrada,Lista) :-  
    open(FicheroEntrada,read,FlujoEntrada),  
    leerFichero(FlujoEntrada,Lista),  
    close(FlujoEntrada),  
    !.
```

```
leerFichero(FlujoEntrada,[]):-  
    at_end_of_stream(FlujoEntrada), !.
```

```
leerFichero(FlujoEntrada,[N|Lista]):-  
    read(FlujoEntrada,N),  
    N \= end_of_file, !,  
    leerFichero(FlujoEntrada,Lista).
```

```
leerFichero(FlujoEntrada,Lista):-  
    read(FlujoEntrada,_),  
    leerFichero (FlujoEntrada,Lista).
```



# Índice

1. Apertura de ficheros y cierre de flujos
2. Lectura y escritura de términos
- 3. Lectura y escritura de caracteres**
4. Modificación de los dispositivos de entrada y salida actuales

### 3. Lectura y escritura de caracteres

- Escritura
- Lectura
- Ejemplo de lectura y escritura de fichero de caracteres

### 3. Lectura y escritura de caracteres

- **Escritura**
- Lectura
- Ejemplo de lectura y escritura de fichero de caracteres

### 3. Lectura y escritura de caracteres

- **Escritura**

- *nl*
- *put, put\_char, put\_code*
- *tab*
- Escritura de cadenas de caracteres

### 3. Lectura y escritura de caracteres

- **Escritura**

- *nl*
- *put, put\_char, put\_code*
- *tab*
- Escritura de cadenas de caracteres

### 3. Lectura y escritura de caracteres

- **Escritura**

- *nl*

- **Sintaxis**

*nl*([Flujo])

- Flujo:

- ✓ Flujo asociado a un fichero abierto para escritura.

### 3. Lectura y escritura de caracteres

- **Escritura**

- *nl*

- **Descripción**

- Escribe un salto de línea en el dispositivo de salida actual (*current\_output*)

- nl*

- El dispositivo de salida por defecto es la pantalla.

- Escribe un salto de línea en el flujo indicado

- nl(Flujo)*

- Se escribirá en el fichero al cerrar el flujo

### 3. Lectura y escritura de caracteres

- **Escritura**

- *nl*

- **Descripción**

- Escribe un **salto de línea** (*new line*).

- Solamente se satisface una vez.

- **Ejemplo**

?- *write(1), nl, write(2).*

*1*

*2*

*true*



### 3. Lectura y escritura de caracteres

- **Escritura**

- *nl*
- ***put, put\_char, put\_code***
- *tab*
- Escritura de cadenas de caracteres

### 3. Lectura y escritura de caracteres

- **Escritura**

- *put, put\_char, put\_code*

- **Sintaxis**

*put*([Flujo,] argumento)

- **Descripción**

- El argumento debe ser

- ✓ un **átomo** con un **carácter**

- ✓ o un **valor numérico** que se corresponda con un **carácter**,

- ✓ o una **cadena** de caracteres con un **carácter**

- Solamente se satisface una vez.

### 3. Lectura y escritura de caracteres

- **Escritura**

- *put, put\_char, put\_code*

- **Ejemplo**

*?- put('L').*

*L*

*true.*

*?- put(76).*

*L*

*true.*

*?- put("L").*

*L*

*true.*

### 3. Lectura y escritura de caracteres

- **Escritura**

- *put, put\_char, put\_code*

- **Ejemplo**

?- *put(104), put(111), put(108), put(97).*

*hola*

*true.*

### 3. Lectura y escritura de caracteres

- **Escritura**

- *nl*
- *put, put\_char, put\_code*
- ***tab***
- Escritura de cadenas de caracteres

### 3. Lectura y escritura de caracteres

- **Escritura**

- ***tab***

- **Sintaxis**

***tab***(*[Flujo,]* argumento)

- **Descripción**

- El argumento debe contener un valor **numérico**
- Escribe el número de **espacios en blanco** indicados por el argumento.
- Solamente se satisface una vez.

### 3. Lectura y escritura de caracteres

- **Escritura**

- ***tab***

- **Equivalencia**

*tab(0):- !.*

*tab(N):- put(32),*

*M is N-1,*

*tab(M).*

### 3. Lectura y escritura de caracteres

- **Escritura**

- ***tab***

- **Ejemplo**

?- ***write(uno), tab(1),write(diez), tab(10),write(fin).***

*uno diez            fin*

*true.*



### 3. Lectura y escritura de caracteres

- **Escritura**

- *nl*
- *put, put\_char, put\_code*
- *tab*
- **Escritura de cadenas de caracteres**

### 3. Lectura y escritura de caracteres

- **Escritura**

- **Escritura de cadenas de caracteres**

**Observación: antiguas versiones de Prolog**

?- *write("Cadena maravillosa").*

*[67,97,100,101,110,97,32,109,97,114,97,118,105  
,108,108,111,115,97]*

*true.*

?- *display("Cadena maravillosa").*

*.(67,.(97,.(100,.(101,.(110,.(97,.(32,.(109,.(97,.(  
114,.(97,.(118,.(105,.(108,.(108,.(111,.(115,.  
(97,[ ])))))))))))))*

*true.*

### 3. Lectura y escritura de caracteres

- **Escritura**

- **Escritura de cadenas de caracteres**

- **Definición**

*escribir\_cadena([]).*

*escribir\_cadena([Cabeza | Cola]):-*

***put**(Cabeza),*

*escribir\_cadena(Cola).*

### 3. Lectura y escritura de caracteres

- **Escritura**

- **Escritura de cadenas de caracteres**

- **Ejemplo**

*?- escribir\_cadena("Cadena maravillosa").*

*Cadena maravillosa*

*true.*

### 3. Lectura y escritura de caracteres

- Escritura
- **Lectura**
- Ejemplo de lectura y escritura de fichero de caracteres

### 3. Lectura y escritura de caracteres

- Lectura

- *get0*
- *get*
- Lectura de una frase y transformación en átomos
- Mostrar por pantalla el contenido de un fichero

### 3. Lectura y escritura de caracteres

- Lectura

- *get0*

- *get*

- Lectura de una frase y transformación en átomos

- Mostrar por pantalla el contenido de un fichero

### 3. Lectura y escritura de caracteres

- Lectura

- *get0*

- Sintaxis

*get0*([Flujo,] Variable)

- Descripción

- Lee el siguiente **carácter** que se teclee

- ✓ Desde el *Flujo* asociado a un fichero de lectura

- ✓ O desde el dispositivo de entrada actual (*current\_input*), que es el teclado, por defecto

- La lectura finaliza al pulsar la tecla de “*enter*”.



### 3. Lectura y escritura de caracteres

- Lectura

- *get0*

- Ejemplos

?- *get0(X), put(X).*

|: *a*



Se pulsa la tecla de “*enter*”  
para finalizar

*a*

*X = 97.*

?- *get0(X), get0(Y), put(X), put(Y).*

|: *ab*

*ab*

*X = 97,*

*Y = 98*

### 3. Lectura y escritura de caracteres

- Lectura

- *get0*

- Ejemplos

?- *get0(X), put(X).*

|:



Se pulsa la tecla de “*enter*”  
para finalizar

*X = 10.*

### 3. Lectura y escritura de caracteres

- Lectura

- *get0*

- ***get***

- Lectura de una frase y transformación en átomos

- Mostrar por pantalla el contenido de un fichero

### 3. Lectura y escritura de caracteres

- Lectura

- *get*

- Sintaxis

*get*([Flujo,] Variable)

- Descripción

- Lee el siguiente carácter **imprimible** que se teclee.
- La lectura finaliza al pulsar la tecla de “*enter*”.

### 3. Lectura y escritura de caracteres

- Lectura

- *get*

- Ejemplos

?- *get(X), put(X).*

|:

|: *b*

*X = 98.*

?- *get(X), get(Y), put(X), put(Y).*

|:

|: *a*

*b*

*ab*

*X = 97,*

*Y = 98.*

Se pulsa la tecla de “*enter*” para finalizar

### 3. Lectura y escritura de caracteres

- Lectura

- Observación

- *Hay más predicados de lectura de caracteres similares*

- *get\_byte, get\_char, get\_code*

- *peek\_byte, peek\_char, peek\_code*

- ...

### 3. Lectura y escritura de caracteres

- **Lectura**

- *get0*

- *get*

- **Lectura de una frase y transformación en átomos**

- **Mostrar por pantalla el contenido de un fichero**

### 3. Lectura y escritura de caracteres

- Lectura

- Lectura de una frase y transformación en átomos

- Primera parte

*leer\_frase(Palabras):-*

*get0(Character),*

*leer\_resto(Character,Palabras).*



### 3. Lectura y escritura de caracteres

- Lectura

- Lectura de una frase y transformación en átomos

- Segunda parte

*/\* El punto "." (carácter 46) indica el fin de la frase \*/*

```
leer_resto(46,[ ]):- !.
```

*/\* Se omite el espacio en blanco (carácter 32) \*/*

```
leer_resto(32,Palabras):- !,
```

```
leer_frase(Palabras).
```

*/\* Lee los caracteres de la palabra actual \*/*

```
leer_resto(Character,[Palabra|Palabras]):-
```

```
leer_caracteres(Character,Caracteres,Siguiente_caracter),
```

```
name(Palabra,Caracteres),
```

```
leer_resto(Siguiente_caracter,Palabras).
```

### 3. Lectura y escritura de caracteres

- **Lectura**

- **Lectura de una frase y transformación en átomos**

- **Observación**

***name**(Palabra, Caracteres)*

- Hace la conversión entre un átomo y una cadena de caracteres.

- **Ejemplos**

*?- name(Palabra, "Cadena").  
Palabra = 'Cadena'.*

*?- name(cadena, Caracteres).*

*Caracteres = [99, 97, 100, 101, 110, 97]*

### 3. Lectura y escritura de caracteres

- Lectura

- Lectura de una frase y transformación en átomos

- Tercera parte

```
/* Fin de palabra: 46 = punto "." */
```

```
leer_caracteres(46,[],46):- !.
```

```
/* Fin de palabra: 32 = espacio en blanco */
```

```
leer_caracteres(32,[],32):- !.
```

```
leer_caracteres(Character,
```

```
    [Character|Caracteres],
```

```
    Siguiente_caracter):-
```

```
    get0(Nuevo_caracter),
```

```
    leer_caracteres(Nuevo_caracter,Caracteres,
```

```
    Siguiente_caracter).
```

### 3. Lectura y escritura de caracteres

- Lectura

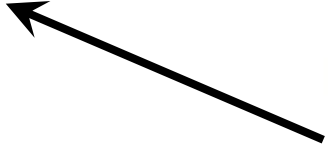
- Lectura de una frase y transformación en átomos

- Ejemplo

?- leer\_frase(X).

| Esta frase va a ser transformada.

X = ['Esta', frase, va, a, ser, transformada]



Se utilizan las comillas simples para que no sea una variable, sino un átomo.

### 3. Lectura y escritura de caracteres

- Lectura

- *get0*

- *get*

- Lectura de una frase y transformación en átomos

- **Mostrar por pantalla el contenido de un fichero**

### 3. Lectura y escritura de caracteres

- Lectura

- Mostrar por pantalla el contenido de un fichero

- Leer\_fichero\_1.pl (1 / 2)

*program*(File):-

*% Open the file for reading*

*open*(File, read, *Stream*),

*% Loop driving by failure*

*repeat*,

*get0*(*Stream*,Character),

*process*(Character),

*% Close*

*close*(*Stream*),

!.

### 3. Lectura y escritura de caracteres

- Lectura

- Mostrar por pantalla el contenido de un fichero

- Leer\_fichero\_1.pl (2 / 2)

- % Character of end of file = -1*

- process(-1):- !.*

- % Process the rest of characters*

- process(Character):- put(Character), **fail**.*

- Ejecución

- ?- [leer\_fichero\_1].*

- ?- browser('nombre.txt').*

### 3. Lectura y escritura de caracteres

- Lectura

- Mostrar por pantalla el contenido de un fichero

- Leer\_fichero\_2.pl (1 / 2)

```
program:- % The name of the file is asked to the user.  
    display("File with quotes-> "), read(String),  
    % Convert the String into an atom: File  
    name(File,String),  
    % Open for reading the file  
    open(File, read, Stream),  
    % Loop driving by failure  
    repeat,  
        get0(Stream,Character),  
        process(Character),  
    % Close  
    close(Stream), !.
```



### 3. Lectura y escritura de caracteres

- Lectura

- Mostrar por pantalla el contenido de un fichero

- Leer\_fichero\_2.pl (2 / 2)

*% Character of end of file = -1*

*process(-1):- !.*

*% Process the rest of characters*

*process(Character):- put(Character), fail.*

- Ejecución

?- [leer\_fichero\_2].

?- browser.

File with double quotes-> "nombre.txt".

Se debe teclear el punto

### 3. Lectura y escritura de caracteres

- Escritura
- Lectura
- **Ejemplo de lectura y escritura de fichero de caracteres**

### 3. Lectura y escritura de caracteres

- Ejemplo de lectura y escritura de fichero de caracteres

- `copy_characters_file.pl`

- copia de un fichero de caracteres (1/3)

`copy`(*InputFile*,*OutputFile*) :-

`open`(*InputFile*,*read*,*InputStream*), *% Open for reading*

`open`(*OutputFile*,*write*,*OutputStream*), *% Open for writing*

`repeat`, *% Loop driving by failure*

`get0`(*InputStream*, *Atom*), *% read the character*

`process`(*OutputStream*,*Atom*), *% process the character*

*% Close the streams*

`close`(*InputStream*),

`close`(*OutputStream*),

`!. % Stop now.`

### 3. Lectura y escritura de caracteres

- Ejemplo de lectura y escritura de fichero de caracteres
  - `copy_characters_file.pl`
    - copia de un fichero de caracteres (2/3)

*% Character of end of file: -1*

*process(\_, -1) :- !.*

*% Process the atom*

*process(OutputStream, Character) :-*

*% Write the character*

*put(OutputStream, Character),*

*% Loop driving by failure*

*fail.*

### 3. Lectura y escritura de caracteres

- Ejemplo de lectura y escritura de fichero de caracteres
  - `copy_characters_file.pl`
    - copia de un fichero de caracteres (3/3)  
?- `copy` (`'entrada.txt'`, `'salida.txt'`).

# Índice

1. Apertura de ficheros y cierre de flujos
2. Lectura y escritura de términos
3. Lectura y escritura de caracteres
4. **Modificación de los dispositivos de entrada y salida actuales**

## 4. Modificación de los dispositivos de entrada y salida actuales

- Primer método:
  - *see, seeing, seen*
  - *tell, telling, told*
- Segundo método:
  - *current\_input, current\_output*
  - *set\_intput, set\_output*

## 4. Modificación de los dispositivos de entrada y salida actuales

- **Primer método:**
  - *see, seeing, seen*
  - *tell, telling, told*
- **Segundo método:**
  - *current\_input, current\_output*
  - *set\_intput, set\_output*



## 4. Modificación de los dispositivos de entrada y salida actuales

- **Primer método:**
  - *see, seeing, seen*
  - *tell, telling, told*

## 4. Modificación de los dispositivos de entrada y salida actuales

- **Primer método:**
  - *see, seeing, seen*
  - *tell, telling, told*

## 4. Modificación de los dispositivos de entrada y salida actuales

- Primer método:

- *see*

- Sintaxis

- see(argumento)*

## 4. Modificación de los dispositivos de entrada y salida actuales

- Primer método:

- *see*

- Descripción (1/2)

- Abre para lectura el fichero indicado por el argumento
      - El fichero pasa a ser el dispositivo de **lectura** actual.
      - Si el argumento es el átomo *user* entonces la lectura se realizará desde el teclado.

## 4. Modificación de los dispositivos de entrada y salida actuales

- Primer método:

- *see*

- Descripción (2/2)

- Si el argumento indica un fichero entonces

- ✓ si **no** estaba **abierto**, la lectura **empieza desde el principio** del fichero.

- ✓ si ya estaba **abierto**, la lectura **continúa desde el punto inmediatamente posterior** a la de la última lectura.

## 4. Modificación de los dispositivos de entrada y salida actuales

- Primer método:

- *see*

- Ejemplos

- Apertura del fichero entrada.txt del directorio /home/usuario

?- *see*('/home/usuario/entada.txt').

- Apertura del fichero indicado por la variable X

?- *read*(X), *see*(X).

## 4. Modificación de los dispositivos de entrada y salida actuales

- Primer método:
  - *see, **seeing**, seen*
  - *tell, telling, told*

## 4. Modificación de los dispositivos de entrada y salida actuales

- Primer método:

- *seeing*

- Sintaxis

*seeing*(argumento)

- Descripción

- ❑ Si argumento es una **variable no instanciada** entonces le **asocia** el nombre del **dispositivo de entrada actual**.
- ❑ Si argumento es una **variable instanciada** o una **constante** entonces se **comprueba** si es el nombre del **dispositivo de entrada actual**.



## 4. Modificación de los dispositivos de entrada y salida actuales

- Primer método:

- *seeing*

- Ejemplos

?-*seeing*('datos').

Es cierto si *datos* es el dispositivo de lectura actual.

?- *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<sup>29</sup>

## 4. Modificación de los dispositivos de entrada y salida actuales

- **Primer método:**
  - *see, seeing, **seen***
  - *tell, telling, told*

## 4. Modificación de los dispositivos de entrada y salida actuales

- Primer método:

- *seen*

- Sintaxis

*seen*

- Descripción

- **Cierra** el fichero de lectura actual, volviendo el teclado (*user*) a ser el dispositivo de lectura actual.

## 4. Modificación de los dispositivos de entrada y salida actuales

- Primer método:

- *see, seen*

- Ejemplos

*/\* 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).*

## 4. Modificación de los dispositivos de entrada y salida actuales

- Primer método:

- *see, seen*

- Ejemplos

```
/* Contenido del fichero entrada.txt */
```

```
agua.
```

```
fuego.
```

```
tierra.
```

```
aire.
```

```
?- see('entrada.txt'), contar(1), seen.
```

```
1 agua
```

```
2 fuego
```

```
3 tierra
```

```
4 aire
```

```
true.
```

## 4. Modificación de los dispositivos de entrada y salida actuales

- **Primer método:**
  - *see, seeing, seen*
  - ***tell***, *telling, told*

## 4. Modificación de los dispositivos de entrada y salida actuales

- Primer método:
  - *tell*
    - Sintaxis

*tell*(argumento)

## 4. Modificación de los dispositivos de entrada y salida actuales

- Primer método:

- *tell*

- Descripción (1/2)

- Abre para escritura el fichero indicado por el argumento
- El fichero pasa a ser el dispositivo de **escritura** actual.
- Si el argumento es el átomo *user* entonces la escritura se realizará en la **pantalla**.



## 4. Modificación de los dispositivos de entrada y salida actuales

- Primer método:

- *tell*

- Descripción (2/2)

- Si el argumento indica un fichero entonces
  - ✓ si no estaba abierto, se abre para escritura.
  - ✓ si ya estaba abierto, la escritura continúa desde el punto inmediatamente posterior al último carácter escrito previamente.

## 4. Modificación de los dispositivos de entrada y salida actuales

- Primer método:

- *tell*

- Ejemplos

- Abre para escritura el fichero salida.txt

- ?- *tell('salida.txt')*.

- Abre para escritura el fichero indicado por *X*

- ?- ..., *tell(X)*.

## 4. Modificación de los dispositivos de entrada y salida actuales

- Primer método:
  - *see, seeing, seen*
  - *tell, **telling**, told*

## 4. Modificación de los dispositivos de entrada y salida actuales

- Primer método:

- *telling*

- Sintaxis

*telling*(argumento)

- Descripción

- ❑ Si argumento es una **variable no instanciada** entonces le **asocia** el nombre del **dispositivo de salida actual**.
- ❑ Si argumento es una **variable instanciada** o una **constante** entonces se **comprueba** si es el nombre del **dispositivo de salida actual**.

## 4. Modificación de los dispositivos de entrada y salida actuales

- Primer método:

- *telling*

- Ejemplos

?- *telling*('datos').

- ❑ Es cierto si *datos* es el dispositivo de salida actual.

?- *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.

## 4. Modificación de los dispositivos de entrada y salida actuales

- Primer método:
  - *see, seeing, seen*
  - *tell, telling, **told***

## 4. Modificación de los dispositivos de entrada y salida actuales

- Primer método:

- *told*

- Sintaxis

*told*

- Descripción

- **Cierra** el fichero asociado al dispositivo de salida actual, volviendo la pantalla (*user*) a ser el dispositivo de salida actual.

## 4. Modificación de los dispositivos de entrada y salida actuales

- Primer método:
  - *see, tell, told, seen*

- Ejemplo

```
/* Contenido del fichero entrada.txt */
```

```
agua.
```

```
fuego.
```

```
tierra.
```

```
aire.
```

```
/* Fin del contenido del fichero */
```

```
?-see('entrada.txt'),tell('salida.txt'),contar(1),told, seen.  
true
```

```
/* Contenido del fichero salida.txt */
```

```
1 agua
```

```
2 fuego
```

```
3 tierra
```

```
4 aire
```



## 4. Modificación de los dispositivos de entrada y salida actuales

- **Primer método:**

- **Ejemplos finales**

- browser\_atoms\_v1.pl
  - Muestra por pantalla los átomos de un fichero
- browser\_atoms\_v2.pl
  - Muestra por pantalla los átomo de un fichero, cuyo nombre se pide desde el teclado.

- Fuente:

[http://www.cpp.edu/~jrfisher/www/prolog\\_tutorial/contents.html](http://www.cpp.edu/~jrfisher/www/prolog_tutorial/contents.html)

## 4. Modificación de los dispositivos de entrada y salida actuales

- **Primer método:**

- **Ejemplos finales**

- **browser\_atoms\_v1.pl**

- Muestra por pantalla los átomos de un fichero

- **browser\_atoms\_v2.pl**

- Muestra por pantalla los átomo de un fichero, cuyo nombre se pide desde el teclado.

- **Fuente:**

[http://www.cpp.edu/~jrfisher/www/prolog\\_tutorial/contents.html](http://www.cpp.edu/~jrfisher/www/prolog_tutorial/contents.html)

## 4. Modificación de los dispositivos de entrada y salida actuales

- Primer método:

- Ejemplos finales

- **browser\_atoms\_v1.pl (1/3)**

*browse*(File) :-

*seeing*(Old), % Save the current input

*see*(File), % open this file

*repeat*, % Loop driving by failure

*read*(Atom), % Read the atom from File

*process*(Atom), % Process the atom

*seen*, % Close File

*see*(Old), % Restore the previous current input

!. % Stop now

## 4. Modificación de los dispositivos de entrada y salida actuales

- Primer método:

- Ejemplos finales

- `browser_atoms_v1.pl` (2/3)

```
process(end_of_file) :- !.
```

```
process(Atom) :- write(Atom), nl, fail.
```

## 4. Modificación de los dispositivos de entrada y salida actuales

- Primer método:

- Ejemplos finales

- `browser_atoms_v1.pl (3/3)`

```
?- [browser_atoms_v1].
```

```
% browser compiled 0.00 sec, 1 clauses
```

```
true.
```

```
?- browse('entrada.txt').
```

```
agua.
```

```
fuego.
```

```
tierra.
```

```
aire.
```

```
true.
```

## 4. Modificación de los dispositivos de entrada y salida actuales

- **Primer método:**

- **Ejemplos finales**

- `browser_atoms_v1.pl`
  - Muestra por pantalla los átomos de un fichero
- `browser_atoms_v2.pl`
  - Muestra por pantalla los átomos de un fichero, cuyo nombre se pide desde el teclado.
- Fuente:

[http://www.cpp.edu/~jrfisher/www/prolog\\_tutorial/contents.html](http://www.cpp.edu/~jrfisher/www/prolog_tutorial/contents.html)

## 4. Modificación de los dispositivos de entrada y salida actuales

- Primer método:

- Ejemplos finales

- **browser\_atoms\_v2.pl (1/3)**

*browse:-*

*% The name of the file is asked to the user.*

*display("File with double quotes-> "), read(String),*

*% Convert the string into an atom: File*

*name(File,String),*

*seeing(Old), % Save for later the current input*

*see(File), % Open the file and set as current input*

*repeat,*

*read(Atom), % Read the atom from file*

*process(Atom),*

*seen, % Close the file*

*see(Old), % The previous current input is restored*

*!. % Stop now*

## 4. Modificación de los dispositivos de entrada y salida actuales

- Primer método:

- Ejemplos finales

- `browser_atoms_v2.pl (2/3)`

- ```
process(end_of_file) :- !.
```

- ```
process(Data):- write(Data), nl, fail.
```



## 4. Modificación de los dispositivos de entrada y salida actuales

- Primer método:

- Ejemplos finales

- **browser\_atoms\_v2.pl (3/3)**

*?- [browser\_interactivo].*

*% browser\_interactivo compiled 0.00 sec, 1 clauses  
true .*

*?- browse.*

*File with double quotes: "entrada.txt".*

*agua*

*fuego*

*tierra*

*aire*

*true.*

## 4. Modificación de los dispositivos de entrada y salida actuales

- Primer método:
  - *see, seeing, seen*
  - *tell, telling, told*
- Segundo método:
  - *current\_input, current\_output*
  - *set\_intput, set\_output*

## 4. Modificación de los dispositivos de entrada y salida actuales

- Segundo método:
  - *current\_input, current\_output*
  - *set\_intput, set\_output*

## 4. Modificación de los dispositivos de entrada y salida actuales

- Segundo método:
  - *current\_input*, *current\_output*
  - *set\_intput*, *set\_output*

## 4. Modificación de los dispositivos de entrada y salida actuales

- Segundo método:

- *current\_input*

- Sintaxis

*current\_input* (Argumento)

- Descripción

- Si Argumento no está instanciado,

- ✓ toma el valor del **dispositivo de entrada actual**

- En caso contrario,

- ✓ se comprueba si el argumento es el **dispositivo de entrada actual**

## 4. Modificación de los dispositivos de entrada y salida actuales

- Segundo método:
  - *current\_input*, *current\_output*
  - *set\_intput*, *set\_output*

## 4. Modificación de los dispositivos de entrada y salida actuales

- Segundo método:

- *current\_output*

- Sintaxis

*current\_output* (Argumento)

- Descripción

- Si Argumento no está instanciado,

- ✓ toma el valor del **dispositivo de salida actual**

- En caso contrario,

- ✓ se comprueba si el argumento es el **dispositivo de salida actual**

## 4. Modificación de los dispositivos de entrada y salida actuales

- Segundo método:
  - *current\_input*, *current\_output*
  - ***set\_intput***, *set\_output*



## 4. Modificación de los dispositivos de entrada y salida actuales

- Segundo método:

- *set\_input*

- Sintaxis

*set\_input* (*Flujo*)

- Descripción

- *Flujo* se convierte en el **dispositivo de entrada actual**

- *Flujo* debe haber sido abierto mediante *open(Fichero,read,*Flujo*)*

## 4. Modificación de los dispositivos de entrada y salida actuales

- Segundo método:
  - *current\_input*, *current\_output*
  - *set\_intput*, *set\_output*

## 4. Modificación de los dispositivos de entrada y salida actuales

- Segundo método:

- *set\_output*

- Sintaxis

*set\_output* (*Flujo*)

- Descripción

- *Flujo* se convierte en el dispositivo de salida actual

- *Flujo* debe haber sido abierto mediante

*open*(*Fichero*,*write*,*Flujo*)

## 4. Modificación de los dispositivos de entrada y salida actuales

- Segundo método:
  - Ejemplos
    - Lectura de átomo desde un fichero y desde el teclado
    - Lectura de átomos desde un fichero
    - Lectura de caracteres desde un fichero
    - Numerar los átomos de un fichero

## 4. Modificación de los dispositivos de entrada y salida actuales

- **Segundo método:**
  - **Ejemplos**
    - **Lectura de átomos desde un fichero y desde el teclado**
    - **Lectura de átomos desde un fichero**
    - **Lectura de caracteres desde un fichero**
    - **Numerar los átomos de un fichero**

## 4. Modificación de los dispositivos de entrada y salida actuales

- Segundo método:
    - Ejemplos: lectura de átomos desde un fichero y desde el teclado
- ?- *open('nuevo.txt',read,X), **set\_input(X)**, read(Dato), write(Dato), **set\_input(user)**, nl, read(Nuevodato), write(Nuevodato), close(X).*

*agua*

|: **mar.**

*mar*

*X = <stream>(0xe63040),*

*Dato = agua,*

*Nuevodato = mar.*

## 4. Modificación de los dispositivos de entrada y salida actuales

- Segundo método:
  - Ejemplos: lectura de átomos desde un fichero y desde el teclado
  - ?- `open('nuevo.txt',read,X), set_input(X), read(Dato), write(Dato), set_input(user), nl, read(Nuevodato), write(Nuevodato), close(X).`

*agua*

|: *mar.*

*mar*

`X = <stream>(0xe63040),`

`Dato = agua,`

`Nuevodato = mar.`

Leído desde el fichero

## 4. Modificación de los dispositivos de entrada y salida actuales

- Segundo método:
  - Ejemplos: lectura de átomos desde un fichero y desde el teclado

?- `open('nuevo.txt',read,X), set_input(X), read(Dato), write(Dato), set_input(user), nl, read(Nuevodato), write(Nuevodato), close(X).`

*agua*

|: *mar.*

*mar*

*X = <stream>(0xe63040),*

*Dato = agua,*

*Nuevodato = mar.*

Leído desde el teclado



## 4. Modificación de los dispositivos de entrada y salida actuales

- **Segundo método:**
  - **Ejemplos**
    - Lectura de átomo desde un fichero y desde el teclado
    - **Lectura de átomos desde un fichero**
    - Lectura de caracteres desde un fichero
    - Numerar los átomos de un fichero

## 4. Modificación de los dispositivos de entrada y salida actuales

- Segundo método:
  - Ejemplos
    - Lectura de átomos desde un fichero
      - browser\_atoms\_v3.pl
        - ✓ Muestra por pantalla los átomos de un fichero
      - browser\_atoms\_v4.pl
        - ✓ Muestra por pantalla los átomos de un fichero, cuyo nombre se pide desde el teclado.

## 4. Modificación de los dispositivos de entrada y salida actuales

- Segundo método:

- Ejemplos

- **browser\_atoms\_v3.pl (1/3)**

```
browse(File) :- % Save for later the current input
```

```
current_input(Old),
```

```
% Open for reading the file
```

```
open(File,read,Stream),
```

```
% The stream is the new current input
```

```
set_input(Stream),
```

```
repeat, % Loop driving by failure
```

```
read(Atom), % read the atom from file
```

```
process(Atom), % process the atom
```

```
close(Stream), % Close the stream
```

```
set_input(Old), %Restore the previous current input
```

```
!. % Stop now
```

## 4. Modificación de los dispositivos de entrada y salida actuales

- Segundo método:
  - Ejemplos
    - `browser_atoms_v3.pl (2/3)`

```
process(end_of_file) :- !.
```

```
process(Atom) :- write(Atom), nl, fail.
```

## 4. Modificación de los dispositivos de entrada y salida actuales

- Segundo método:
  - Ejemplos
    - `browser_atoms_v3.pl (3/3)`

```
?- [browser_atoms_v3].
```

```
% browser compiled 0.00 sec, 1 clauses
```

```
true.
```

```
?- browse('entrada.txt').
```

```
agua.
```

```
fuego.
```

```
tierra.
```

```
aire.
```

```
true.
```

## 4. Modificación de los dispositivos de entrada y salida actuales

- Segundo método:
  - Ejemplos
    - Lectura de átomos desde un fichero
      - browser\_atoms\_v3.pl
        - ✓ Muestra por pantalla los átomos de un fichero
      - browser\_atoms\_v4.pl
        - ✓ Muestra por pantalla los átomos de un fichero, cuyo nombre se pide desde el teclado.

## 4. Modificación de los dispositivos de entrada y salida actuales

- Segundo método:

- Ejemplos

- `browser_atoms_v4.pl (1/3)`

```
browse :- % The name of the file is asked to the user.  
display("File with double quotes-> "), read(String),  
name(File,String), % Convert the string into an atom: File  
current_input(Old), % Save for later the current input  
open(File,read,Stream), % Open for reading the file  
set_input(Stream), % The stream is the new current input  
repeat, % Loop driving by failure  
    read(Atom), % read the atom from file  
    process(Atom), % process the atom  
    close(Stream), % Close the stream  
    set_input(Old), %Restore the previous current input  
!. % Stop now
```

## 4. Modificación de los dispositivos de entrada y salida actuales

- Segundo método:
  - Ejemplos
    - `browser_atoms_v4.pl (2/3)`

```
process(end_of_file) :- !.
```

```
process(Data):- write(Data), nl, fail.
```



## 4. Modificación de los dispositivos de entrada y salida actuales

- Segundo método:

- Ejemplos

- **browser\_atoms\_v4.pl (3/3)**

```
?- [browser_atoms_v4].
```

```
% browser_interactivo compiled 0.00 sec, 1 clauses  
true .
```

```
?- browse.
```

```
File with quotes: "entrada.txt".
```

```
agua
```

```
fuego
```

```
tierra
```

```
aire
```

```
true.
```

## 4. Modificación de los dispositivos de entrada y salida actuales

- Segundo método:
  - Ejemplos
    - Lectura de átomo desde un fichero y desde el teclado
    - Lectura de átomos desde un fichero
    - **Lectura de caracteres desde un fichero**
    - Numerar los átomos de un fichero

## 4. Modificación de los dispositivos de entrada y salida actuales

- Segundo método:
  - Ejemplos
    - Lectura de átomos desde un fichero
      - `browser_character_v1.pl`
        - ✓ Muestra por pantalla el contenido de un fichero: lee y escribe caracteres
      - `browser_character_v2.pl`
        - ✓ Muestra por pantalla el contenido de un fichero, cuyo nombre se pide desde el teclado.

## 4. Modificación de los dispositivos de entrada y salida actuales

- Segundo método:
  - Ejemplos: browser\_character\_v1.pl(1/2)

*browser\_character\_1*(File):-

```
% Save the current input for later  
current_input(Old),  
% Open the File indicated by the Atom  
open(File,read,Stream),  
% Stream is the current input device  
set_input(Stream),  
% Loop driving by failure  
repeat,  
    get0(Character),  
    process(Character),  
% The Stream is closed  
close(Stream),  
!.
```

Leído desde el fichero

## 4. Modificación de los dispositivos de entrada y salida actuales

- Segundo método:
  - Ejemplos: `browser_character_v1.pl(2/2)`

*% End of file character = -1*

*process(-1):- !.*

*% Process the rest of characters*

*process(Character):- put(Character), fail.*

## 4. Modificación de los dispositivos de entrada y salida actuales

- Segundo método:
  - Ejemplos: browser\_character\_v2.pl (1/2)

```
browser_character_2:- % The name of the file is asked to the user.  
    display("File with double quotes-> "), read(String),  
    % Convert the String into an Atom  
    name(File,String),  
    % Save for later the current input  
    current_input(Old),  
    % Open the File indicated by the Atom  
    open(File,read,Stream),  
    % Stream is the current input device  
    set_input(Stream),  
    % Loop driving by failure  
    repeat,  
        get0(Character),  
        process(Character),  
    % The Stream is closed  
    close(Stream),  
    !.
```

Leído desde el fichero



## 4. Modificación de los dispositivos de entrada y salida actuales

- Segundo método:
  - Ejemplos: browser\_character\_v2.pl (2/2)

*% End of file character = -1*

*process(-1):- !.*

*% Process the rest of characters*

*process(Character):- put(Character), fail.*

## 4. Modificación de los dispositivos de entrada y salida actuales

- **Segundo método:**
  - **Ejemplos**
    - Lectura de átomo desde un fichero y desde el teclado
    - Lectura de átomos desde un fichero
    - Lectura de caracteres desde un fichero
    - Numerar los átomos de un fichero



## 4. Modificación de los dispositivos de entrada y salida actuales

- Segundo método:
  - Ejemplos: numerar los átomos de un fichero

*% Se numeran los elementos leídos a partir de N*

*contar(N):-*

*read(Termino),*

*mostrar(Termino,N).*

*% Atom of end of file*

*mostrar(end\_of\_file,\_):- !.*

*mostrar(Termino,N):-*

*write(N),*

*tab(2),*

*write(Termino),*

*nl,*

*N1 is N + 1,*

*contar(N1).*

## 4. Modificación de los dispositivos de entrada y salida actuales

- Segundo método:
  - Ejemplos: numerar los átomos de un fichero

*program*(File):-

*% Save for later the previous current input device*

*current\_input*(Old),

*% Open the file and set as current input*

*open*(File,read,Stream),

*set\_input*(Stream),

*% Process the file*

*contar*(1),

*% Close the stream*

*close*(Stream),

*% Restore the previous current input device*

*set\_input*(Old),

*!. % Stop now*

## 4. Modificación de los dispositivos de entrada y salida actuales

- Segundo método:
  - Ejemplos: numerar los átomos de un fichero

?- *program*('entrada.txt').

1 *agua*

2 *fuego*

3 *tierra*

4 *aire*

*true.*



UNIVERSIDAD DE CÓRDOBA

ESCUELA POLITÉCNICA SUPERIOR DE CÓRDOBA

DEPARTAMENTO DE  
INFORMÁTICA Y ANÁLISIS NUMÉRICO



# PROGRAMACIÓN DECLARATIVA

INGENIERÍA INFORMÁTICA

CUARTO CURSO

PRIMER CUATRIMESTRE

**Tema 12.- Entrada y salida**

