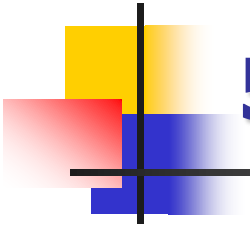


5. Entrada y Salida en PROLOG

1. Escritura de términos
2. Lectura de términos
3. Escritura y lectura de caracteres
4. Escritura en fichero
5. Lectura de fichero

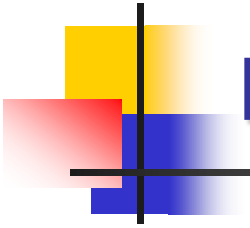
1. Escritura de términos (I)

- Predicado predefinido "**write(X)**":
  - Siempre se satisface
  - Si X está instanciada, se muestra en pantalla.
  - Si no, se saca la variable interna ('_G244').
  - Nunca se intenta re-satisfacer.

```
?- write(beatriz).
beatriz
Yes
?- write('Beatriz').
Beatriz
Yes
```

```
?- write(Beatriz).
_G244
Beatriz = _G244 ;
No
?- write("Beatriz").
[66, 101, 97, 116, 114, 105, 122]
Yes
```

Escritura de términos (II)



- "**nl**" provoca un salto de línea. (?- nl.)
- "**tab(X)**" escribe X espacios en blanco.
  - No coincide con la función de una máquina de escribir convencional.
- Ambas sólo se satisfacen una sola vez.
- "**display(X)**" → muestra X sin interpretar los functores/operadores.

Escritura de términos (III)

?- nl, write('Beatriz'), tab(5), write('Gonzalez'), nl, write('Francisco'), tab(5), write('Perez').

Beatriz Gonzalez
Francisco Perez

Yes

?- L = [1, 2, 3, 4, 5], write(L).
[1, 2, 3, 4, 5]

L = [1, 2, 3, 4, 5]
yes
?- L = [1, 2, 3, 4, 5], display(L).
.(1, .(2, .(3, .(4, .(5, [])))))

L = [1, 2, 3, 4, 5]
yes

Escritura de términos (IV)

```
?- write(a+b*c/d), nl, display(a+b*c/d), nl.
a+b*c/d
+(a, /(*(b, c), d))
Yes
```

- Por ejemplo, el operador "+" es declarado infijo con dos argumentos.

- "display" puede ser útil para recordar el orden de precedencia:

```
?- write((a+b)*c/d), nl, display((a+b)*c/d), nl.
(a+b)*c/d
/(*(+(a, b), c), d)
Yes
```

2. Lectura de términos (I)

- "read(X)", lee por teclado un término, que se instanciará a la variable X.

- El término debe ir seguido de "." y un carácter no imprimible como el espacio o el retorno de carro.

```
hola :- write('Nombre: '), read(Nombre),
    write('Primer Apellido: '), read(Apellido1),
    write('Segundo Apellido: '), read(Apellido2), nl,
    write('Hola '), write(Nombre), tab(1),
    write(Apellido1), tab(1), write(Apellido2).

?- hola.
Nombre: Carlos.
Primer Apellido: Alonso.
Segundo Apellido: Gonzalez.

Hola _L131 _L132 _L133        (Lee términos: funciona con minúsculas sólo)
```

Lectura de términos (II)

?- hola.
Nombre: carlos.
Primer Apellido: alonso.
Segundo Apellido: gonzalez.

Hola carlos alonso gonzalez

?- hola.
Nombre: 'Carlos'.
Primer Apellido: 'Alonso'.
Segundo Apellido: 'Gonzalez'.

Hola Carlos Alonso Gonzalez

- Las mayúsculas se utilizan para designar a variables, por eso se mostraba el valor del puntero a estas supuestas variables: Carlos, Alonso y Gonzalez.
- Podemos introducir términos en minúsculas, o cadenas.

Lectura de términos (III)

- **¿Qué ocurre si ponemos?**

?- hola.

Nombre: “Carlos”.

Primer Apellido: “Alonso”.

Segundo Apellido: “Gonzalez”.

Hola ??

Hola [67, 97, 114, 108, 111, 115] [65, 108, 111, 110, 115, 111] [71, 111, 110, 122, 97, 108, 101, 122]

3. Escritura y lectura de caracteres (I)

- El carácter es la unidad más pequeña objeto de lectura o escritura.

- Si una variable está instanciada a un código ASCII (entero positivo):
  "**put(X)**" escribe el correspondiente carácter.
- No admite resatisfacción.

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

?- X = 104, Y = 111, Z = 108, W=97, put(X), put(Y),
   put(Z), put(W).
hola.
```

Escritura y lectura de caracteres (II)

- "**get0(X)**" y "**get(X)**" son objetivos que se satisfacen si X no está instanciada.
- Si X está instanciada, intenta hacer *matching* con entrada.
- Además, no se resatisfacen.

- Diferencia: "get0(X)" espera a instanciar en X el primer carácter tecleado, mientras que "get(X)" captura el primero imprimible.

```
?- get(X).
|:<CR>
|: .
X = 46
```

```
?- get0(X).
|: <CR>

X = 10 ;
```

Ejercicios: L/E de términos

- Ejemplo: escritura estructurada de listas.
  - La complejidad de la escritura de listas se debe al posible anidamiento con otras listas o estructuras.
  - muestra(X, I): "muestra términos lista"
    - X debe instanciar a la lista que se quiere mostrar
    - I espacios sangría inicial
    - ?- muestra([1, 2, 3], 1).

      1
      2
      3
    - ?- muestra([1,2,[3,4], 5, 6], 6).

      1
      2
        3
        4
      5
      6

4. Escritura en fichero (I)

- Existe un *Canal de salida activo*: por defecto, la pantalla.
- "tell(X)", establece la salida según lo instanciado por X.
- Es un predicado que requiere que X esté instanciada; en caso contrario, se produce un error.
- Como objetivo, no permite la resatisfacción.

Escritura en fichero (II)

- Ejercicio:
  - Desviar la salida del predicado “muestra” hacia el fichero c:\tmp\salida.txt

    ?- tell('c:/tmp/salida.txt').

    ?- muestra([1, 2, [3, 4], 5, 6], -3).

    ?- told.

- Este último predicado cierra el fichero y restablece la pantalla como canal de salida.

Escritura en fichero (III)

- Para averiguar el canal activo, se emplea el predicado "**telling(X)**". Aquí X está por instanciar.

```
?- telling(X).
X = user ;
No

?- tell('d:/tmp/salida.prolog').
Yes

?- telling(X).
X = '$stream'(112102) ;
No

?- told.
Yes
```

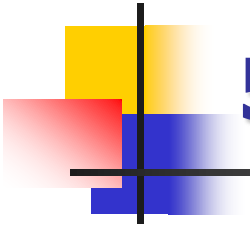
Escritura en fichero (IV)

- Para volver a la pantalla como canal activo, sin cerrar el fichero abierto previamente, se emplea: “tell(user)”.

```
?- tell('c:/tmp/pp1.txt'),
muestra([1, 2, [3, 4], 5, 6], -3),
tell(user), muestra([1, 2, [3, 4], 5, 6]),
tell('c:/tmp/pp2.txt'),
muestra([7, [8, [9, [0]]]], 0),
told, muestra([7, [8, [9, [0]]]], 0), told.
```

  - Ejercicio: insertar “telling” en esta secuencia para averiguar el canal activo.

5. Lectura de ficheros (I)



- Las mismas consideraciones, sólo que:
  - El canal de entrada por defecto será el teclado (user).
  - El predicado "tell" se sustituye por "see":
    - ?- see(X).
    - ?- seeing(X).
    - ?- seen.
- Si la lectura se efectúa para ampliar la base de conocimiento, se empleará "consult".

Lectura de ficheros (II)

- Ejemplo: construir una base de conocimiento fruto de la unión otras tres.

```
?- consult('muestra').
% muestra compiled 0.00 sec, 40 bytes

?- consult('campeones').
% campeones compiled 0.00 sec, 80 bytes

?- consult('densidad').
% Densidad compiled 0.02 sec, 60 bytes


?- ['muestra', 'campeones', 'densidad'].
% muestra compiled 0.00 sec, 40 bytes
% campeones compiled 0.00 sec, 80 bytes
% Densidad compiled 0.02 sec, 60 bytes
```