Tema 10:
Clips: Lectura de datos

Lectura de datos

- Sesión

```
CLIPS> (clear)
CLIPS> (defrule lectura
         =>
         (printout t " Escribe un color: " crlf)
         (assert (color (read))))
CLIPS> (reset)
CLIPS> (run)
 Escribe un color:
verde
CLIPS> (facts)
f-0     (initial-fact)
f-1     (color verde)
For a total of 2 facts.
CLIPS>
```

- Comentarios
  - `(read)`

Lectura y prueba (I)

- Fichero `ej-1.clp` (I)

```
(defglobal
  ?*numero* = 7)

(defrule juego
  =>
  (assert (lee)))

(defrule lee
  ?h <- (lee)
  =>
  (retract ?h)
  (printout t "Escribe un numero: ")
  (assert (numero (read))))

(defrule bajo
  ?h <- (numero ?n)
  (test (< ?n ?*numero*))
  =>
  (retract ?h)
  (printout t ?n " es bajo" crlf)
  (assert (lee)))

(defrule alto
  ?h <- (numero ?n)
  (test (> ?n ?*numero*))
  =>
  (retract ?h)
  (printout t ?n " es alto" crlf)
  (assert (lee)))
```

Lectura y prueba (I)

- **Fichero `ej-1.clp` (II)**

```
(defrule exacto
  ?h <- (numero ?n)
  (test (= ?n ?*numero*))
  =>
  (retract ?h)
  (printout t ?n " es correcto" crlf))
```

- **Sesión**

```
CLIPS> (clear)
CLIPS> (load "ej-1.clp")
CLIPS> Defining defglobal: numero
Defining defrule: juego +j
Defining defrule: lee +j
Defining defrule: bajo +j
Defining defrule: alto +j
Defining defrule: exacto +j
TRUE
CLIPS> (reset)
CLIPS> (run)
Escribe un numero: 3
3 es bajo
Escribe un numero: 9
9 es alto
Escribe un numero: 7
7 es correcto
CLIPS>
```

Lectura y restricciones (I)

- Fichero `ej-2.clp` (I)

```
(defglobal
  ?*numero* = 7)

(defrule juego
  =>
  (assert (lee)))

(defrule lee
  ?h <- (lee)
  =>
  (retract ?h)
  (printout t "Escribe un numero: ")
  (assert (numero (read))))

(defrule bajo
  ?h <- (numero ?n&:(< ?n ?*numero*))
  =>
  (retract ?h)
  (printout t ?n " es bajo" crlf)
  (assert (lee)))

(defrule alto
  ?h <- (numero ?n&:(> ?n ?*numero*))
  =>
  (retract ?h)
  (printout t ?n " es alto" crlf)
  (assert (lee)))
```

Lectura y restricciones (II)

- Fichero `ej-2.clp` (II)

```
(defrule exacto
  ?h <- (numero ?n&:(= ?n ?*numero*))
  =>
  (retract ?h)
  (printout t ?n " es correcto" crlf))
```

- Sesión

```
CLIPS> (clear)
CLIPS> (load "ej-2.clp")
CLIPS> Defining defglobal: numero
Defining defrule: juego +j
Defining defrule: lee +j
Defining defrule: bajo +j
Defining defrule: alto +j
Defining defrule: exacto +j
TRUE
CLIPS> (reset)
CLIPS> (run)
Escribe un numero: 3
3 es bajo
Escribe un numero: 9
9 es alto
Escribe un numero: 7
7 es correcto
CLIPS>
```

Lectura de hechos como cadenas

- Sesión

```
CLIPS> (defrule inserta-hecho
         =>
         (printout t "Escribe un hecho como cadena"
                     crlf)
         (assert-string (read)))
CLIPS> (reset)
CLIPS> (run)
Escribe un hecho como cadena
"(color verde)"
CLIPS> (facts)
f-0     (initial-fact)
f-1     (color verde)
For a total of 2 facts.
CLIPS>
```

- Comentarios
  - (assert-string <cadena>)

Lectura de líneas

- Sesión

```
CLIPS> (defrule lee-linea
         =>
         (printout t "Introduce datos." crlf)
         (bind ?cadena (readline))
         (assert-string (str-cat "(" ?cadena ")")))
CLIPS> (reset)
CLIPS> (run)
Introduce datos.
colores verde azul ambar rojo
CLIPS> (facts)
f-0     (initial-fact)
f-1     (colores verde azul ambar rojo)
For a total of 2 facts.
CLIPS>
```

- Comentarios
  - `(readline)`
  - `(str-cat <cadena>*)`

Sumario de funciones Clips utilizadas

- `read`
- `readline`
- `assert-string`
- `str-cat`