

Clips: Funciones

José A. Alonso y Francisco J. Martín

Ciencias de la Computación e Inteligencia Artificial

UNIVERSIDAD DE SEVILLA

Funciones de cadenas

● Ejemplos

```
CLIPS> (str-cat "ej-" 17 ".txt")
"ej-17.txt"
CLIPS> (sym-cat "ej-" 17 ".txt")
ej-17.txt
CLIPS> (sub-string 4 5 "ej-17.txt")
"17"
CLIPS> (str-index "17" "ej-17.txt")
4
CLIPS> (eval "(+ 3 4)")
7
CLIPS> (build "(defrule r-1 (a) => (assert (b)))")
TRUE
CLIPS> (ppdefrule r-1)
(defrule MAIN::r-1
  (a)
  =>
  (assert (b)))
CLIPS> (upcase "ej-17.txt")
"EJ-17.TXT"
CLIPS> (lowcase "EJ-17.TXT")
"ej-17.txt"
CLIPS> (str-compare "ab" "ab")
0
CLIPS> (str-compare "ab" "bcd")
-1
CLIPS> (str-compare "bcd" "ab")
1
CLIPS> (str-length "ej-17.txt")
9
```

Funciones de cadenas

- **Sintaxis**

`(str-cat <expresión>*)`

`(sym-cat <expresión>*)`

`(sub-string <numero-entero> <numero-entero> <cadena>)`

`(str-index <lexema> <lexema>
<lexema> ::= <símbolo> | <cadena>`

`(eval <lexema>)`

`(build <lexema>)`

`(upcase <lexema>)`

`(lowcase <lexema>)`

`(str-compare <lexema> <lexema>)`

`(str-length <lexema>)`

- **Ayuda**

`(help) // FUNCTION_SUMMARY // STRING_FUNCTIONS`

Funciones de campos múltiples

● Ejemplos

```
CLIPS> (create$ a (+ 2 3) "Bc")
(a 5 "Bc")
CLIPS> (nth 2 (create$ a (+ 2 3) "Bc"))
5
CLIPS> (member b (create$ a b c))
2
CLIPS> (member d (create$ a b c))
FALSE
CLIPS> (subsetp (create$ a b b c) (create$ b c a))
TRUE
CLIPS> (subsetp (create$ a d) (create$ a c))
FALSE
CLIPS> (delete$ (create$ a b c d e f) 2 4)
(a e f)
CLIPS> (explode$ "a b c d")
(a b c d)
CLIPS> (implode$ (create$ a b c d))
"a b c d"
CLIPS> (subseq$ (create$ a b c d e f) 2 4)
(b c d)
CLIPS> (replace$ (create$ a b c d e) 3 4 x y z)
(a b x y z e)
CLIPS> (insert$ (create$ a b c d) 3 x y)
(a b x y c d)
CLIPS> (first$ (create$ a b c))
(a)
CLIPS> (rest$ (create$ a b c))
(b c)
```

Funciones de campos múltiples

- **Sintaxis**

```
(create$ <expresión>*)  
(nth$ <numero-entero> <multicampo>)  
(member$ <expresión> <multicampo>)  
(subsetp <multicampo> <multicampo>)  
(delete$ <multicampo> <número-entero> <número-entero>)  
(explode$ <cadena>)  
(implode$ <multicampo>)  
(subseq$ <multicampo> <número-entero> <número-entero>)  
(replace$ <multicampo> <número-entero> <número-entero>  
  <simple-o-multicampo>+)  
(insert$ <multicampo> <número-entero> <simple-o-multicampo>+)  
(first$ <multicampo>)  
(rest$ <multicampo>)  
(length$ <multicampo>)
```

- **Ayuda**

```
(help) // FUNCTION_SUMMARY // MULTIFIELD_FUNCTIONS
```

Funciones matemáticas

- **Funciones matemáticas básicas**

```
(+ <expresión-numérica> <expresión-numérica>+)  
(- <expresión-numérica> <expresión-numérica>+)  
(* <expresión-numérica> <expresión-numérica>+)  
(/ <expresión-numérica> <expresión-numérica>+)  
(div <expresión-numérica> <expresión-numérica>+)  
(max <expresión-numérica>+)  
(min <expresión-numérica>+)  
(abs <expresión-numérica>)  
(float <expresión-numérica>)  
(integer <expresión-numérica>)
```

- **Ayuda**

```
FUNCTION_SUMMARY // MATH_FUNCTIONS // BASIC_MATH_FUNCTIONS
```

Funciones matemáticas

- **Funciones trigonométricas**

acos	arco coseno
acosh	arco coseno hiperbólico
acot	arco tangente
acoth	arco tangente hiperbólico
acsc	arco secante
acsch	arco secant hiperbólico
asec	arco secant
asech	arco secante hiperbólico
asin	arco seno
asinh	arco seno hiperbólico
atan	arco tangente
atanh	arco tangente hiperbólico
cos	coseno
cosh	coseno hiperbólico
cot	cotangente
coth	tangente hiperbólica
csc	cosecante
csch	cosecante hiperbólica
sec	secante
sech	secante hiperbólica
sin	seno
sinh	seno hiperbólico
tan	tangente
tanh	tangente hiperbólica

- **Ayuda**

```
FUNCTION_SUMMARY // MATH_FUNCTIONS // TRIGONOMETRIC_FUNCTIONS
```

Funciones matemáticas

- **Otras funciones matemáticas**

```
(deg-grad <expresión-numérica>)  
(deg-rad <expresión-numérica>)  
(grad-deg <expresión-numérica>)  
(rad-deg <expresión-numérica>)  
(pi)  
(sqrt <expresión-numérica>)  
(** <expresión-numérica> <expresión-numérica>)  
(exp <expresión-numérica>)  
(log <expresión-numérica>)  
(log10 <expresión-numérica>)  
(round <expresión-numérica>)  
(mod <expresión-numérica> <expresión-numérica>)
```

- **Ayuda**

```
FUNCTION_SUMMARY // MATH_FUNCTIONS // EXTENDED_MATH_FUNCTIONS
```


Otras funciones

- **Ejemplos**

```
CLIPS> (gensym)
gen1
CLIPS> (gensym)
gen2
CLIPS> (setgen 23)
23
CLIPS> (gensym)
gen23
CLIPS> (random)
16838
CLIPS> (length "El_Pais")
7
```

- **Sintaxis**

```
(gensym)
(gensym*)
(setgen <número-entero>)
(random)
(length <lexema-o-multicampo>)
```

- **Ayuda**

```
FUNCTION_SUMMARY // MISCELLANEOUS_FUNCTIONS
```

Ordenes de entorno

- **Sintaxis**

```
(load <nombre-de-fichero>)  
(save <nombre-de-fichero>)  
(load <nombre-de-fichero>)  
(bsave <nombre-de-fichero>)  
(clear)  
(exit)  
(reset)  
(batch <nombre-de-fichero>)  
(system <lexema>*)  
(apropos <lexema>)
```

- **Ayuda**

```
COMMAND_SUMMARY // ENVIRONMENT_COMMANDS
```

Ordenes de depuración

- **Sintaxis**

```
(dribble-on <nombre-de-fichero>)  
(dribble-off)  
(watch <item-de-vigilancia>  
  <item-de-vigilancia>  
  ::= all |  
        compilations |  
        statistics |  
        rules <nombre-de-regla>* |  
        activations <nombre-de-regla>* |  
        facts <nombre-de-plantilla>*)  
(unwatch <item-de-vigilancia>)  
(list-watch-items [<item-de-vigilancia>])
```

- **Ayuda**

```
COMMAND_SUMMARY // DEBUGGING_COMMANDS
```

Entrada y salida

- notas.txt

Juan Jose Aguado Casas	5
Rafael Aguila Puente	2
Carlos Alba Adame	9
Maria Luisa Alcalde Perez	3

- aprobado.txt

Juan Jose Aguado Casas	5
Carlos Alba Adame	9

Entrada y salida

```
(defrule abrir-ficheros
=>
  (open "notas.txt" notas "r")
  (open "aprobado.txt" aprobados "w")
  (assert (fase lectura-de-notas)))

(defrule lee-notas
  ?fase <- (fase lectura-de-notas)
=>
  (retract ?fase)
  (bind ?linea (readline notas))
  (if (eq ?linea EOF)
      then (assert (fase cerrar-ficheros))
      else (assert (nota (explode$ ?linea))))))

(defrule escribe-aprobados
  ?nota <- (nota $?nombre ?puntos ?valor)
=>
  (retract ?nota)
  (if (>= ?valor 5)
      then (printout aprobados (implode$ ?nombre)
                                " " ?puntos
                                " " ?valor crlf))
  (assert (fase lectura-de-notas)))

(defrule cerrar-ficheros
  ?fase <- (fase cerrar-ficheros)
=>
  (retract ?fase)
  (close notas)
  (close aprobados))
```

Entrada y salida

- **Funciones**

```
(open <nombre-fichero> <nombre-logico> [<modo>])  
  <modo> ::= "r" | "w" | "r+" | "a"
```

```
(close [<nombre-logico>])
```

```
(printout <nombre-lógico> <expresión>*)
```

```
(read [<nombre-lógico>])
```

```
(readline [<nombre-lógico>])
```

```
(format <nombre-lógico> <cadena-de-control> <expresión>*)
```

```
(rename <nombre-fichero-viejo> <nombre-fichero-nuevo>)
```

```
(remove <nombre-fichero>)
```

- **Ayuda**

```
(help) // FUNCTION_SUMMARY // IO_FUNCTIONS
```

Ficheros de hechos

● Ejemplo

```
CLIPS> (assert (nota 3) (nota 5) (curso 4))
<Fact-2>
CLIPS> (save-facts "z1")
TRUE
CLIPS> (clear)
CLIPS> (load-facts "z1")
TRUE
CLIPS> (facts)
f-0      (nota 3)
f-1      (nota 5)
f-2      (curso 4)
For a total of 3 facts.
CLIPS> (save-facts "z2" visible nota)
TRUE
CLIPS> (clear)
CLIPS> (load-facts "z2")
TRUE
CLIPS> (facts)
f-0      (nota 3)
f-1      (nota 5)
For a total of 2 facts.
```

● Sintaxis

```
(load-facts <nombre-de-fichero>)
```

```
(save-facts <nombre-de-fichero> [<tipo-de-copia> <plantilla>
  <tipo-de-copia> ::= visible | local
```