Navegue por el glosario usando este índice.

A

AND

La función AND es utilizada para evaluar múltiples expresiones y devolver NIL si alguna de ellas es falsa, o la última expresión evaluada si todas son verdaderas. Su sintaxis es la siguiente:

```
(AND expresion1 expresion2 ... expresionN)
```

Donde "expresion1", "expresion2", ..., "expresionN" son las expresiones que se desean evaluar. La función devuelve `NIL` si alguna de las expresiones es falsa, o la última expresión evaluada si todas son verdaderas.

Por ejemplo, para evaluar si los números 2, 4 y 6 son todos pares, se puede utilizar la siguiente expresión:

```
(AND (EVENP 2) (EVENP 4) (EVENP 6))
```

Esto devuelve `T`, ya que todas las expresiones son verdaderas.

Palabra(s) clave: logica

APPEND

La función APPEND es utilizada para concatenar dos o más listas en una sola lista. Su sintaxis es la siguiente:

```
(APPEND &rest listas)
```

Donde "listas" es cualquier número de listas que se deseen concatenar. La función devuelve una nueva lista que contiene todos los elementos de las listas especificadas, en el orden en que se especificaron.

Por ejemplo, para concatenar las listas (1 2) y (3 4) en una sola lista, se puede utilizar la siguiente expresión:

```
(APPEND '(1 2) '(3 4))
```

Esto devuelve la lista (1 2 3 4), que contiene todos los elementos de ambas listas concatenados en el orden en que se especificaron.

Palabra(s) clave: funciones constructoras

Á

Átomo

Los átomos son estructuras básicas en Lisp. Un átomo es un elemento indivisible que tiene significado propio. Pueden ser números (enteros, racionales, complejos, etc.), cadenas de caracteres, o caracteres especiales. Los siguientes son algunos ejemplos válidos de átomos:

- nombre
- 12308907
- *hola*
- abc123

B

BUTLAST

La función BUTLAST es utilizada para obtener todos los elementos de una lista excepto el último. Su sintaxis es la siguiente:

```
(BUTLAST lista n)
```

Donde "lista" es la lista de la cual se quieren obtener los elementos, y "n" es un valor opcional que indica cuántos elementos de la lista se deben excluir, empezando por el último. Si "n" no se especifica, se excluye solo el último elemento.

Por ejemplo, para obtener todos los elementos de la lista (1 2 3 4) excepto el último, se puede utilizar la siguiente expresión:

```
(BUTLAST '(1 2 3 4))
```

Esto devuelve la lista (1 2 3), que son todos los elementos de la lista excepto el último.

Esto no modifica el contenido de la lista original. Si quisiéramos que el resultado se asigne a una nueva lista, haríamos:

```
(SETQ lista (BUTLAST lista n))
```

C

CAR

La función CAR, devuelve el primer elemento de una lista. Su sintaxis es (CAR LISTA), o de una manera más técnica sería, (CAR CONS).

Ejemplo:

```
> (CAR '(1 2 3)) ==> 1
> (CAR ‘(A B C) ) ==> A
> (CAR ‘(A B (C D) E) ) ==> A
> (CAR ‘((A B) C) ) ==> (A B)
> (CAR ‘((A B C)) ) ==> (A B C)
> (CAR ‘A) ==> ERROR
```

No se puede obtener el CAR de un ÁTOMO. En general no se puede aplicar ninguna función de LISTA a los ATOMOS. Es importante destacar que la función CAR no genera un nuevo valor, sino que da como resultado la dirección de un elemento que existe.

CDR

La función CDR, para listas propias, de verdad o no punteadas, devuelve la lista sin el primer elemento. La sintaxis de esta función es (CDR LISTA) o de forma técnica (CDR CONS).

```
> (CDR ‘(A B C) ) ==> (B C)
> (CDR ‘(A B (C D) E) ) ==> (B (C D) E)
> (CDR ‘((A B) C) ) ==> (C)
> (CDR ‘((A B C)) ) ==> NIL
> (CDR ‘A) ==> ERROR
```

Los argumentos del CDR deben ser del tipo CONS. Para una lista punteada, el CDR de una estructura CONS puede ser un átomo. Por tanto, en estos casos se devolverá un átomo y no una estructura CONS.

```
> (CDR ‘(A . B) ) ==> B
```

CEILING

Su sintaxis es

(ceiling (expresion o numero)

Redondea un numero decimal o fracción hacia el siguiente numero entero, por ej:

(ceiling 2.5) => 3

(ceiling (/ 3 4)) => 1

Palabra(s) clave: Matematica, aritmetica, redondeo

COND

La función COND es una construcción especial que se utiliza para evaluar una serie de expresiones condicionales. Su sintaxis es la siguiente:

```
(COND
  (condicion1 expresion1)
  (condicion2 expresion2)
  ...
  (condicionN expresionN)
)
```

Donde "condicion1", "condicion2", ..., "condicionN" son expresiones condicionales y "expresion1", "expresion2", ..., "expresionN" son las expresiones que se deben evaluar si la condición correspondiente es verdadera.

Cada expresión condicional es una lista formada por dos elementos: una condición y una expresión. Si la condición es verdadera, se evalúa la expresión correspondiente y se devuelve su valor. Si la condición es falsa, se pasa a la siguiente expresión condicional, y así sucesivamente hasta que se encuentra una condición verdadera, o se llega al final de la lista de expresiones condicionales.

Si todas las condiciones son falsas, la función `COND` devuelve `NIL`.

Por ejemplo, para evaluar el signo de un número, se puede utilizar la siguiente expresión:

```
(COND
  ((> num 0) 'positivo)
  ((< num 0) 'negativo)
  (T 'cero)
)
```

En este caso, la primera expresión condicional evalúa si el número es mayor que cero, la segunda evalúa si es menor que cero, y la tercera evalúa si es igual a cero. Si ninguna de las dos primeras es verdadera, se devuelve la expresión "cero".

Palabra(s) clave: condicionales

CONS

La función CONS es utilizada para crear una nueva lista a partir de dos elementos: el primero (denominado "car") y el segundo (denominado "cdr"). Su sintaxis es la siguiente:

```
(CONS car cdr)
```

Donde "car" es el primer elemento de la nueva lista, y "cdr" es el segundo elemento de la nueva lista. La función devuelve una nueva lista que comienza con "car" y continúa con "cdr".

Por ejemplo, para crear una lista que contenga los elementos 1 y 2, se puede utilizar la siguiente expresión:

```
(CONS 1 (CONS 2 NIL))
```

Esto crea una nueva lista que comienza con 1 y continúa con una lista que comienza con 2 y termina con NIL.

La función CONS puede utilizarse para también para realizar las siguientes operaciones:

Función CONS	*Ejemplo*	*Resultado*	*Sirve para...*
(CONS átomo lista)	(CONS '1 '(2 3 4))	(1 2 3 4)	Añadir un elemento a una lista
(CONS átomo átomo)	(CONS '1'2)	(1 . 2)	Crear una lista impropia
(CONS lista lista)	(CONS '(1 2) '(3 4))	((1 2) 3 4)	Añadir una sublista a una lista
(CONS lista átomo)	(CONS '(1 2)'4)	((1 2) . 4)	Añadir sublista y crear lista impropia

Palabra(s) clave: funciones constructoras

D

DEFUN

La función DEFUN se utiliza para definir una nueva función. Su sintaxis es la siguiente:

```
(DEFUN nombre (argumentos) cuerpo-de-la-funcion)
```

Donde "nombre" es el nombre de la función que se desea definir, "argumentos" es una lista de argumentos que la función acepta, y "cuerpo-de-la-funcion" es una secuencia de expresiones que definen lo que la función hace. La función devuelve el nombre de la función definida.

Por ejemplo, para definir una función que suma dos números, se puede utilizar la siguiente expresión:

```
(DEFUN suma (x y)
  (+ x y))
```

Esto define una función llamada "suma" que acepta dos argumentos, "x" e "y", y devuelve su suma.

Contactar con el soporte del sitio

Usted se ha identificado como Alexis Olivarez (Cerrar sesión)
Resumen de retención de datos
Descargar la app para dispositivos móviles
Cambiar al tema estándar

Desarrollado por Moodle