

# Función miembro de lista, longitud de lista y término n de lista - LISP

## Member of a list, list length and Nth term of a list – LISP

Autor: Luis Fernando Martínez Muñoz

IS&C, Universidad Tecnológica de Pereira, Pereira, Colombia

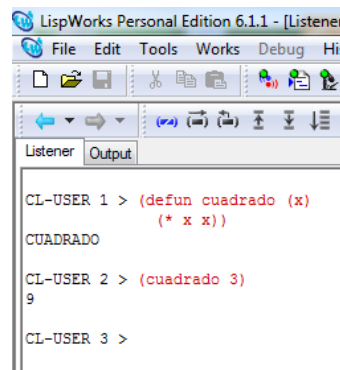
Correo-e: [f.martinez@utp.edu.co](mailto:f.martinez@utp.edu.co)

**Resumen**— El presente artículo se ha escrito con el propósito de ilustrar la implementación de las funciones miembro de lista, longitud de lista y término n de lista escritas en lenguaje LISP.

**Palabras clave**— LISP, listas, función, recursividad.

**Abstract**— This article has been written with the purpose of illustrating the implementation of the member, list length and Nth term of a list functions written in LISP language.

**Key Word**— LISP, lists, function, recursion.



```

LispWorks Personal Edition 6.1.1 - [Listener]
File Edit Tools Works Debug His
[Icons]
Listener Output
CL-USER 1 > (defun cuadrado (x)
              (* x x))
CUADRADO
CL-USER 2 > (cuadrado 3)
9
CL-USER 3 >
  
```

## I. INTRODUCCIÓN

Vamos a desarrollar en este paper, un ensayo sobre las funciones miembro de una lista. Longitud de lista y término n de una lista, su significado y su implementación en lenguaje LISP.

## II. CONTENIDO

Para empezar, veremos algunos elementos del entorno de desarrollo LispWorks, después describiremos las funciones a tratar.

### A. Presentación

Para empezar, utilizaremos el entorno LispWorks, y para ello procederemos a su instalación y posterior uso. El primer paso consistió en instalar la versión para Windows. En los siguientes diagramas presentamos las interfaces generadas.

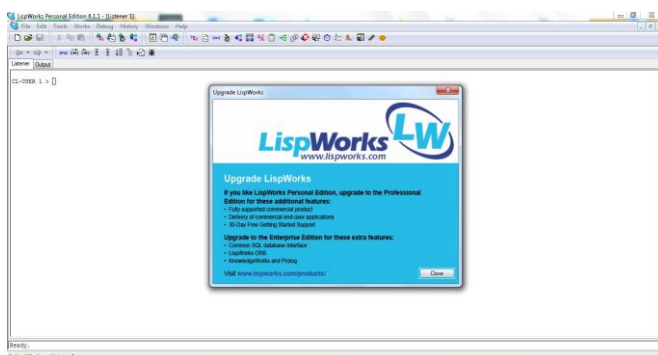


Figura 1. Interfaz de usuario

### B. Fundamento teórico

La recursividad se define como el proceso que para lograr sus propósitos debe llamarse a sí misma.

Se presentan 3 ejemplos como lo son las funciones, miembro de una lista, longitud de lista y término n de una lista.

Veremos en el siguiente apartado como se implantan en LISP.

### C. Ejemplos de aplicación

Función miembro de lista.

```
CL-USER 1 > (defun miembro (objeto lista)
  (if (null lista)
      nil
      (if (eql (car lista) objeto)
          lista
          (miembro objeto (cdr lista))
      )
  )
)
MIEMBRO
CL-USER 2 > (print (miembro 'c '(a b c d)))
(C D)
(C D)
CL-USER 3 > (print (miembro 'h '(a b c d)))
NIL
NIL
```

Figura 2. Función miembro de lista

La función toma como parámetros, un elemento y una lista.

Se tiene como condición de parada que la lista esté vacía, en tal caso, devuelve NIL.

Se compara la cabeza de la lista con el elemento, si coinciden, se devuelve la lista desde donde se halló por primera vez al elemento dentro de ella, de lo contrario se hace un llamado a la función con el elemento y a la cola de la lista como nueva lista.

Si no se encuentra al elemento dentro de la lista, la función devuelve NIL como se describe en la función de parada.

Función longitud de lista.

```
CL-USER 7 : 1 > (defun longitud (lista)
  (cond ((null lista) 0)
        (t (+ (longitud (cdr lista)) 1))))
LONGITUD
CL-USER 8 : 1 > (print (longitud '(a b c d e f g h)))
8
8
CL-USER 9 : 1 >
```

Figura 3. Función longitud de lista

La función toma como parámetro una lista.

Se tiene como condición de parada que la lista esté vacía, en tal caso, devuelve 0 (cero).

Se suma 1 con un llamado a la función con la cola como nueva lista.

Función término n de una lista.

```
CL-USER 9 : 1 > (defun termino-n (n lista)
  (cond ((zerop n) (car lista))
        ; La función zerop verifica si n es cero
        (t (termino-n (- n 1) (cdr lista)))))
TERMINO-N
CL-USER 10 : 1 > (print (termino-n 5 '(a b c d e f g h i j k l)))
F
F
```

Figura 4. Función término n de una lista

La función toma como parámetros, una posición y una lista.

Se tiene como condición de parada que la posición sea 0 (cero), en tal caso, devuelve la cabeza de la lista.

Se hace un llamado a la función con posición -1 y la cola de la lista, de esta forma, cuando se haya restado hasta llegar a 0 (cero) se tendrá el término n de la lista.

## D. Conclusión

La recursividad es un elemento clave para la Inteligencia Artificial. Esta se puede implementar para resolver múltiples problemas con un enfoque más creativo y matemático.

## REFERENCIAS

### Referencias en la Web:

[http://www.redesep.com/materias/blanda/4\\_3\\_recursividad.php](http://www.redesep.com/materias/blanda/4_3_recursividad.php)

[http://www.redesep.com/materias/blanda/5\\_2\\_estructuras\\_recursivas.php](http://www.redesep.com/materias/blanda/5_2_estructuras_recursivas.php)

