

# Funciones - LISP - 01

## Functions – LISP - 01

Autor: Jhonatan Ospina Osorio

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

Correo-e: Jhonatan.ospina@utp.edu.co

**Resumen**— En este documento procederemos a ver una selección de ejercicios trabajados en el lenguaje LISP usando la herramienta LispWorks, desde el manejo de números y ecuaciones hasta manejo de listas.

**Palabras clave**— LISP, LispWorks, Recursividad, Listas.

**Abstract**— in this document we will proceed to see a selection of exercises worked in the LISP language using the LispWorks tool, from the handling of numbers and equations to handling of lists.

**Key Word**— LISP, LispWorks, Recursion, Lists.

### I. INTRODUCCIÓN

A continuación, veremos siete funciones básicas de LISP, unas de tratamiento de números y otras de tratamiento de listas, las que observaremos son las siguientes:

- Valor absoluto.
- Exponenciales.
- Factorial.
- Longitud de una lista.
- Miembro de una lista.
- Sumatoria de n.
- Terminó-n.

### II. CONTENIDO

Los programas en LISP se deben desarrollar en entornos integrados (IDE). El entorno utilizado es LispWorks. Vamos a desarrollar la temática a continuación

LispWorks es una herramienta para trabajar con LISP, permite el uso de breakpoints, macros y muestra las definiciones de manera óptima además de contar con una consola de comandos en directo.

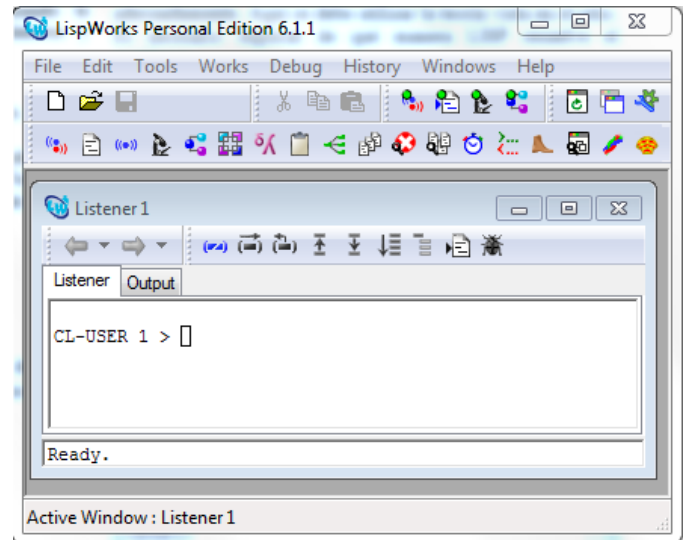


Figura 1. LispWorks Personal Edition 6.1.1.

#### A. Fundamento teórico

Gracias a LispWorks podemos compilar código LISP sin necesidad de ingresar a internet, además, permite un mejor control de los proyectos que trabajemos y podremos usarlo de forma pesada para trabajos de manejo de datos grandes. A continuación, veremos cada una de las funciones que queremos resolver usando LISP en la plataforma LispWorks.

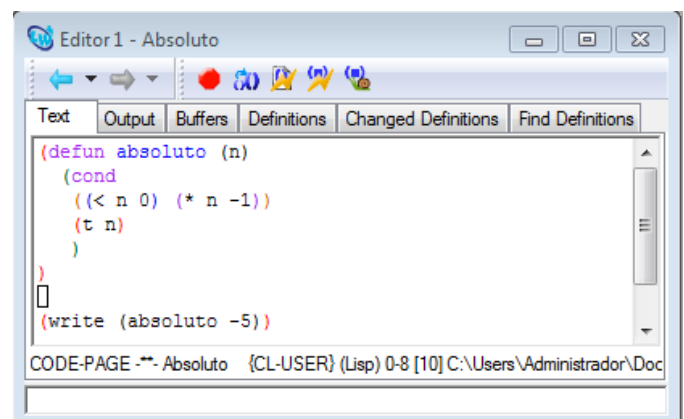
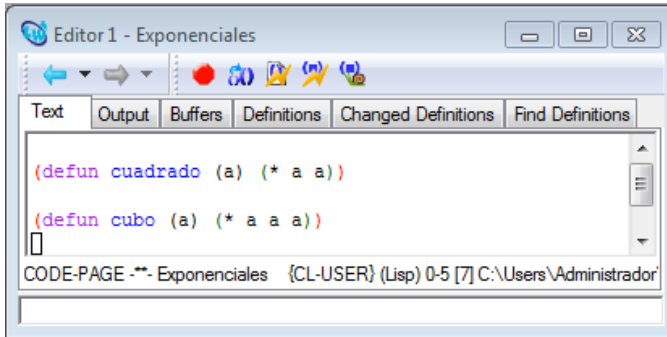


Figura 2. Función valor absoluto.

Esta función nos permite obtener el valor absoluto de un número  $n$ , esta función ya está predefinida en LISP y se puede usar como *(abs n)*.



```

(defun cuadrado (a) (* a a))

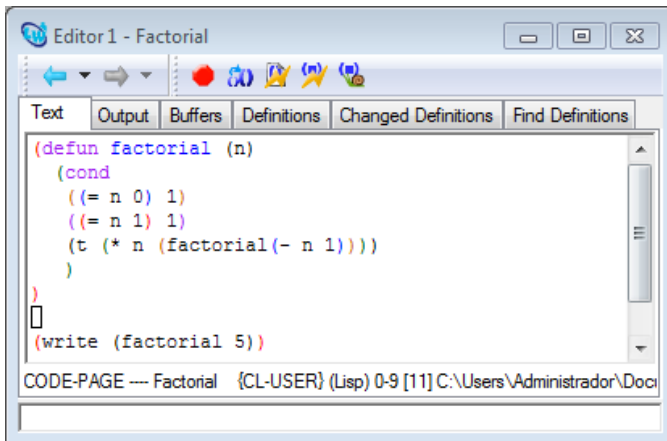
(defun cubo (a) (* a a a))

```

CODE-PAGE -- Exponenciales {CL-USER} (Lisp) 0-5 [7] C:\Users\Administrador\Docu

Figura 3. Función exponencial.

En el código de exponencial tenemos la función cuadrado y la función cubo, para mayores exponentes se pueden usar combinaciones de estos dos.



```

(defun factorial (n)
  (cond
    ((= n 0) 1)
    ((= n 1) 1)
    (t (* n (factorial (- n 1)))))
  )
)

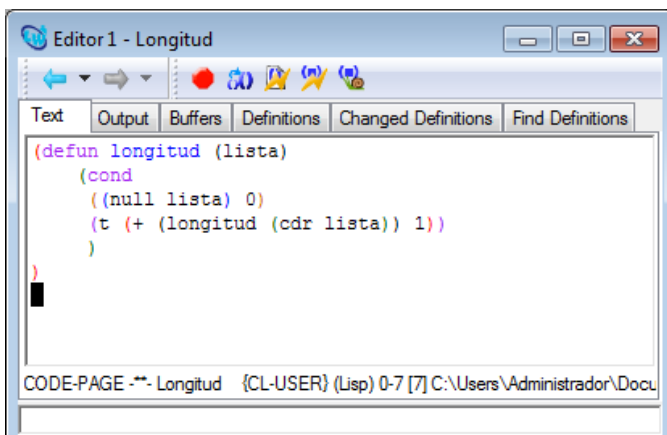
(write (factorial 5))

```

CODE-PAGE --- Factorial {CL-USER} (Lisp) 0-9 [11] C:\Users\Administrador\Docu

Figura 4. Función factorial.

Esta es una de las funciones de más complejidad y mayor tiempo de ejecución de las que tenemos a lo que se refiere en tratamiento de números.



```

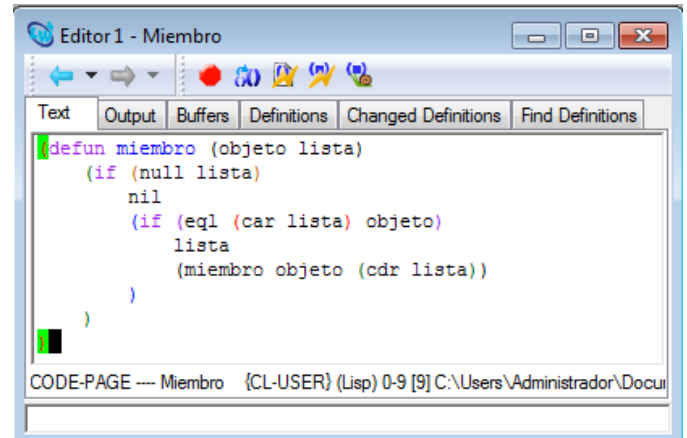
(defun longitud (lista)
  (cond
    ((null lista) 0)
    (t (+ (longitud (cdr lista)) 1))
  )
)

```

CODE-PAGE -- Longitud {CL-USER} (Lisp) 0-7 [7] C:\Users\Administrador\Docu

Figura 5. Función longitud.

La función longitud nos regresa el tamaño de la lista que le ingresemos, además de esto, esta función ya existe dentro de las funciones predefinidas de LISP.



```

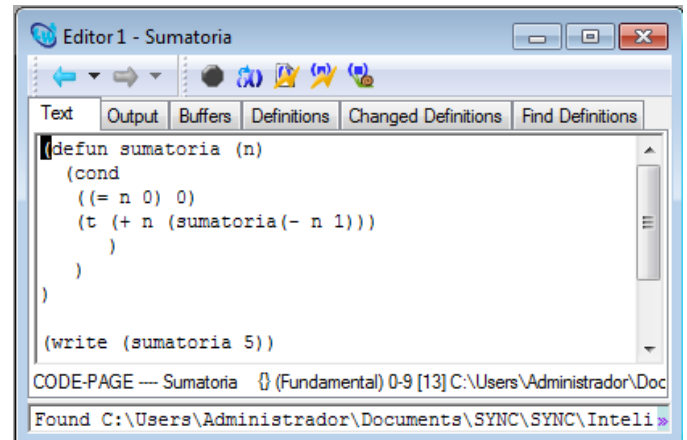
(defun miembro (objeto lista)
  (if (null lista)
      nil
      (if (eql (car lista) objeto)
          lista
          (miembro objeto (cdr lista))
      )
  )
)

```

CODE-PAGE --- Miembro {CL-USER} (Lisp) 0-9 [9] C:\Users\Administrador\Docu

Figura 6. Función miembro.

Esta función nos devuelve un objeto de la lista si existe, podemos ver que se usa una función predeterminada de LISP como lo es *(eql a b)* que nos dice si dos entradas son iguales.



```

(defun sumatoria (n)
  (cond
    ((= n 0) 0)
    (t (+ n (sumatoria (- n 1)))))
  )
)

(write (sumatoria 5))

```

CODE-PAGE --- Sumatoria {} (Fundamental) 0-9 [13] C:\Users\Administrador\Docu

Found C:\Users\Administrador\Documents\SYNC\SYNC\Inteli

Figura 7. Función sumatoria.

La función sumatoria de forma recursiva calcula el resultado de sumar todos los números que preceden al que ingresemos.

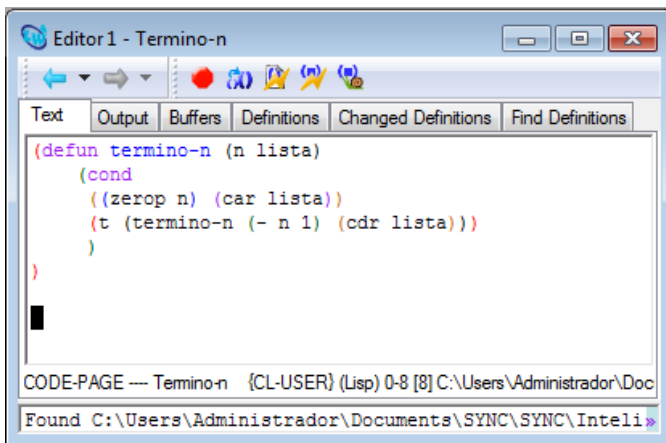


Figura 8. Función termino-n.

Esta función recorre la lista hasta encontrar el objeto de la lista en la posición que le entreguemos, podemos ver que usa una función predefinida de LISP (`zerop n`) que nos determina si `n` es cero.

## B. Conclusión

Podemos ver que gracias a LispWorks es más fácil controlar nuestros archivos y ver su funcionamiento en tiempo real, sin necesidad de depender de herramientas externas.

## REFERENCIAS

### Referencias en la Web:

- [1] <http://riotorto.users.sourceforge.net/Lisp/funciones.html>
- [2] <http://www.lispworks.com/downloads/index.html>