Trabajo con listas - LISP

Working with lists – LISP - 01

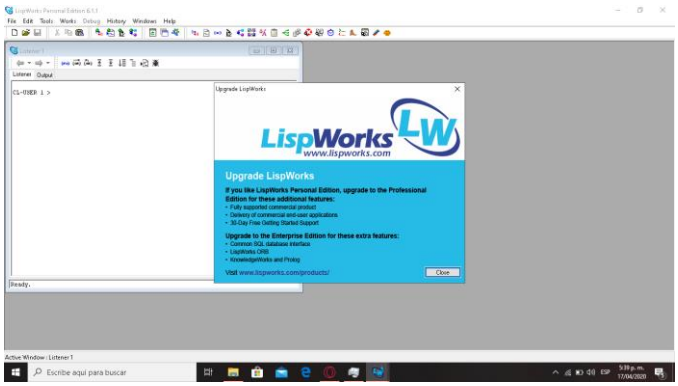
Autor: Esteban Sanchez Lopez
IS&C, Universidad Tecnológica de Pereira, Pereira, Colombia
Correo-e: esteban.sanchez@utp.edu.co

***Resumen*— En este documento veremos el funcionamiento de varias funciones con listas en el lenguaje LISP, describiendo su fundamentación y ejemplos visuales de los resultados de las funciones.**

***Palabras clave*— Listas, Funciones, LISP.**

***Abstract*— In this document we will see the operation of various functions with lists in the LISP language, describing their rationale and visual examples of the results of the functions.**

***Key Word*— Lists, Functions, LISP.**



I. INTRODUCCIÓN

En este artículo se presentan un conjunto de funciones de LISP usando listas, desarrollando las primeras funciones propuestas para realizar un acercamiento al lenguaje.

II. CONTENIDO

Los programas y códigos para el lenguaje LISP van a ser llevados a cabo en el entorno de desarrollo (IDE) llamando LispWorks. Durante el desarrollo del documento se va a realizar los siguientes ejercicios:

- miembro-de (member)
- longitud (length)
- término-n (n-term)
- Algunos otros.

A. Presentación

El IDE elegido para realizar el desarrollo es LispWorks, posee una interfaz sencilla, fácil de entender, además de ser bastante liviano. Cuando se inicia queda completamente lista para empezar a trabajar.

B. Fundamento teórico

Las funciones realizadas a describir son:

- Miembro de (Member)

Esta función recibe el total de 2 argumentos y se representa de la siguiente manera:

- member <item> <list>; donde <item>es el elemento a buscar y <list> la lista en la cual se buscará el ítem.

Esta función tiene 2 retornos posibles:

- en caso de que el elemento no se encuentre en la lista, se retorna nil.
- En caso de encontrarlo, retornara la cola restante desde el punto donde encontró el elemento.

También existe una variante de esta que que se expresa:

Member-if: el cual en lugar de buscar un elemento específico, busca un tipo de dato, por lo cual cuando encuentre un elemento que cumple con el tipo de dato que se busca, devuelve la cola a partir de ese elemento. En caso de no encontrarlo retorna nil.

- Longitud (Length)

Esta función recibe el total de 1 argumento y se representa de la siguiente manera::

- length <list>; donde <list> es la lista en a la cual se le medirá la longitud.

Esta función retorna un número entero que indicará el número de elementos que posee dicha lista.

- Término-n (n-term o nth)

Esta función recibe el total de 2 argumento y se representa de la siguiente manera::

- nth <n> <list>; donde <n> es la posición del elemento en la lista y <list> es la lista en a la cual se buscará el elemento.

Esta función retorna 2 posible valores:

- en caso de dar un número <n> mayor a la longitud de la lista se retornara un nil.
- Si el valor es menor que la longitud de la lista, retorna el elemento en esa posición de la lista.

Otras Funciones:
Ahora vamos a ver algunas funciones que trabajamos en clase anteriormente y usaremos el mismo IDE para ver la el resultado.

Las funciones que veremos son:

- Factorial de N:

La función recibe solo un argumento, el cual es el número al que se quiere calcular su factorial.
Es una función recursiva, por lo que se llama a sí mismo hasta llegar al valor de cero y luego se devuelve a los valores que deja almacenados y va resolviendo hasta llegar al resultado.

- Sumatoria de N:

La función recibe solo un argumento, el cual es el número al que se quiere calcular su sumatoria..
Es una función recursiva, por lo que se llama a sí mismo hasta llegar al valor de cero y luego se devuelve a los valores que deja almacenados y va resolviendo hasta llegar al resultado
Es bastante similar a la sumatoria, pero en lugar de ir multiplicando, realiza suma.

- Valor absoluto de N:

La función recibe un solo argumento, el cual es el número al que se le quiere aplicar el valor absoluto.
La función realiza una condición, en donde si el número ingresado es menor a cero, se retorna la multiplicación del número por menos uno, pero si es mayor se retorna el mismo número sin modificaciones.

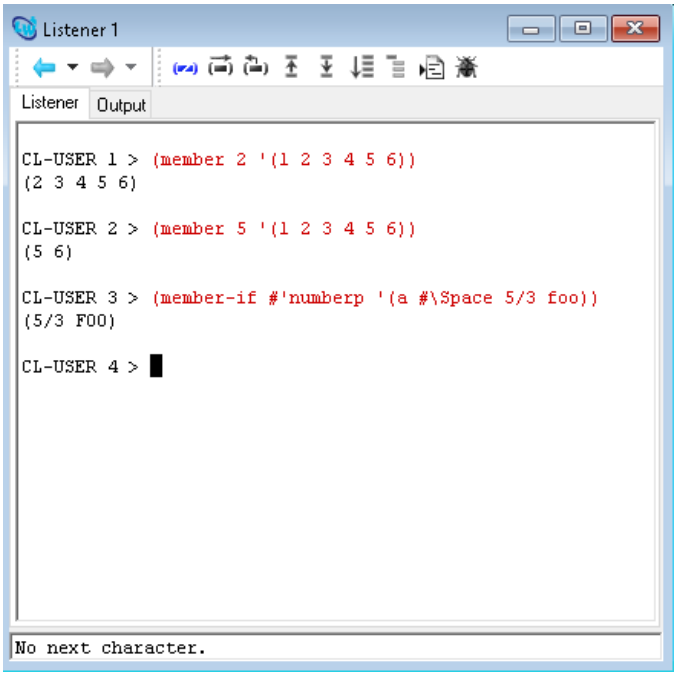


Figura 1. Pruebas de código member usando la función integrada en LispWorks.

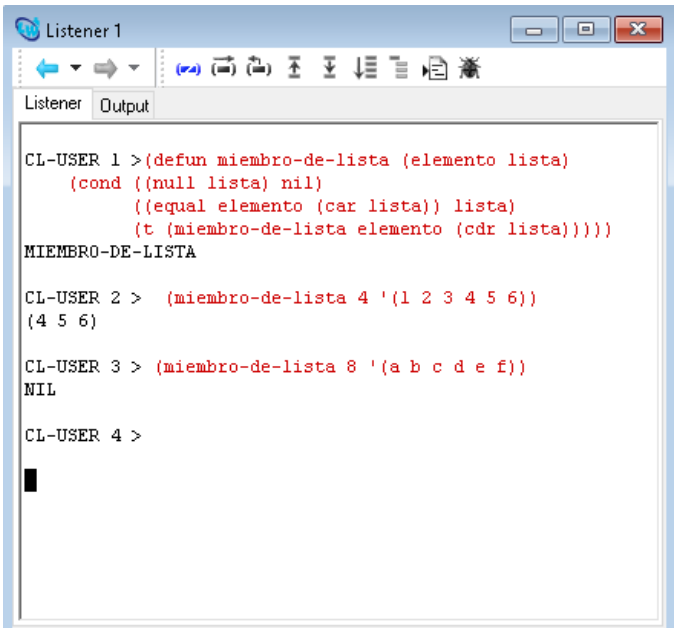


Figura 2. Pruebas de código member implementada manualmente en LispWorks.

C. Ejemplos de aplicación

- Realizando algunos ejemplos de member y member-if:

- Realizando algunos ejemplos de length:

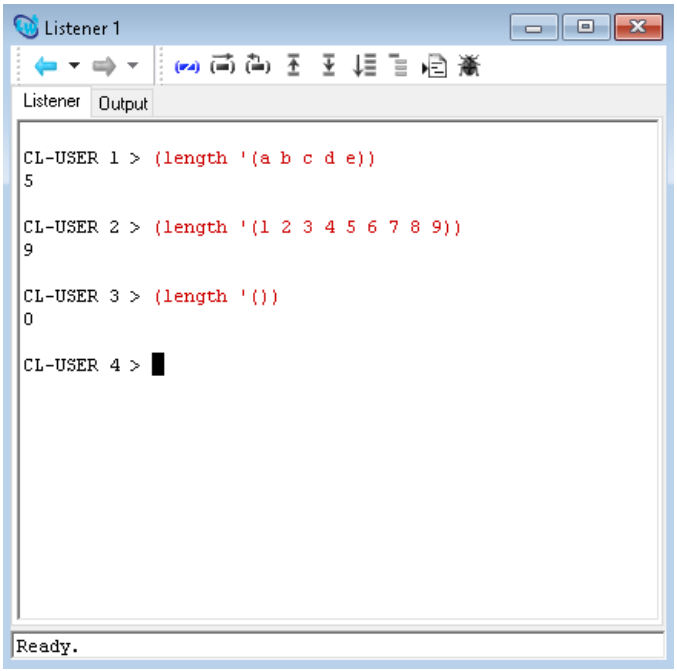


Figura 3. Pruebas de código length usando la función integrada en LispWorks.

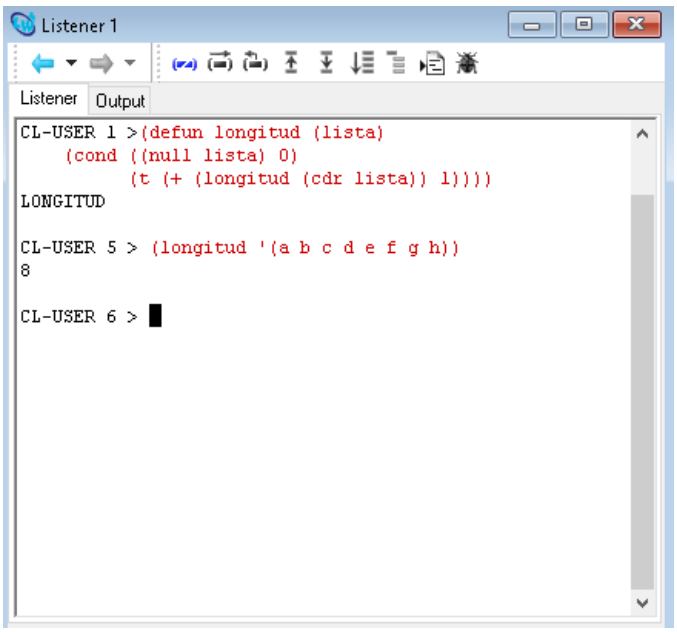


Figura 4. Pruebas de código length implementada manualmente en LispWorks.

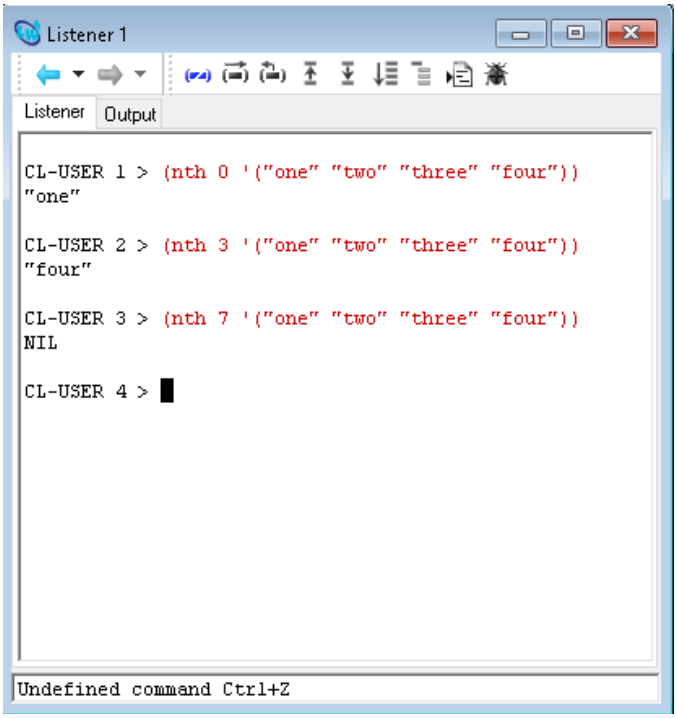


Figura 5. Pruebas de código n-term usando la función integrada en LispWorks.

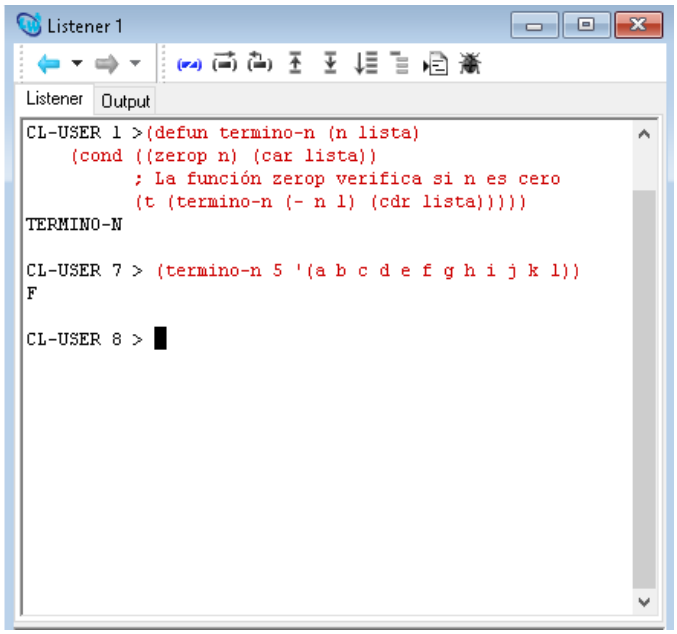


Figura 6. Pruebas de código n-term implementada manualmente en LispWorks.

- Realizando algunos ejemplos de n-term:

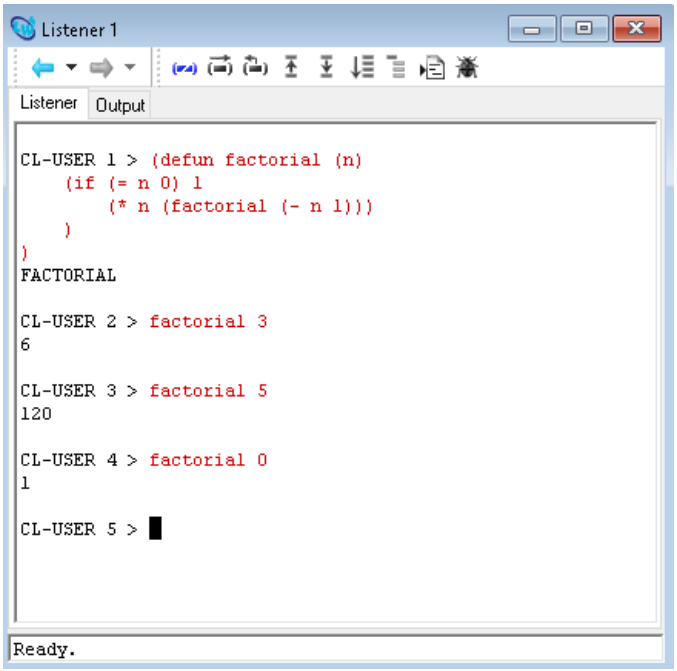


Figura 7. Pruebas de código Factorial..

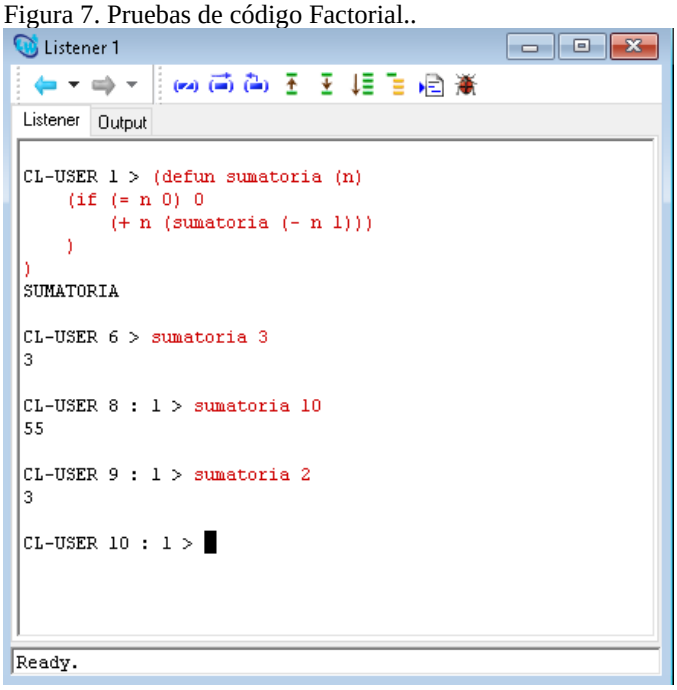


Figura 8. Pruebas de código sumatoria.

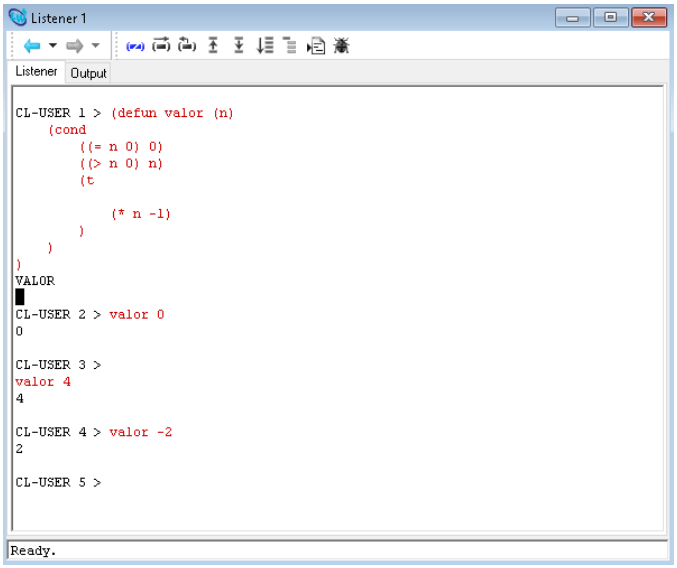


Figura 9. Pruebas de código Valor Absoluto.

D. Conclusión

LISP posee funciones bastante útiles a la hora de realizar manejo de listas, que nos permiten evitar bastantes líneas de código, comparando las funciones implementadas manualmente con las integradas podemos darnos cuenta del ahorro de significativo.

REFERENCIAS

A Continuación aparecen las direcciones de los lugares de los cuales se usó o verificó información usada en el desarrollo del documento.

Referencias en la Web:

[1]http://www.redesep.com/materias/inteligencia/5_2_estructuras_recursivas.php