Tratamiento de listas - LISP

List processing – LISP

Autor: Daniel Patiño Rojas
Estudiante de Ingeniería en Sistemas y computación, Universidad Tecnológica de Pereira, Pereira, Colombia
Correo-e: danielrojas0198@utp.edu.co

***Resumen*— El presente artículo se ha escrito con el propósito de documentar como usar la recursividad en LISP para realizar ciertas funciones tales como buscar un miembro en una lista, cuál es su longitud y buscar el termino N de una lista.**

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

***Abstract*— This article has been written for the purpose of documenting how to use recursion in LISP to perform certain functions such as searching for a member in a list, what is its length and searching for the N term of a list.**

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

I. INTRODUCCIÓN

Este paper está desarrollado con el fin de entender cómo usar la recursividad en el lenguaje LISP, el cual es un lenguaje multiparadigma con una gran historia. El lenguaje de programación LISP fue diseñado originalmente en 1958 por John McCarthy y sus colaboradores en el insti Tecnológico de Massachusetts; siendo este el según lenguaje de programación de alto nivel con mayor antigüe entre los que existen actualmente. LISP, originalmente fue creado en base al cálculo lambda Alonzo Churcha, convirtiéndose así en uno de los lengu favoritos para la investigación de la Inteligencia Artificial. LISP es pionero de muchas ideas en la ciencia de computación tales como las estructuras de datos de árbol manejo de almacenamiento dinámico entre otras. Las li encadenadas son una de las estructuras de datos importa de LISP, dado que su código está compuesto de listas da resultado que los programas escritos en este pue manipularse como si fuera una estructura de datos.

II. CONTENIDO

Para empezar a describir las tres funciones recursivas, explicare en que consiste la recursividad. La recursividad consiste en funciones que se llaman a sí mismas, evitando el uso de bucles e iteradores. A continuación, se describen las siguientes funciones:

A. Función miembro de lista

La función miembro de lista recibe dos componentes: elemento y lista. La función devuelve NIL en caso de no encontrar el elemento en la lista. Si lo encuentra, devuelve la sablista a partir de donde encontró el elemento.

```
; Definición de la función miembro-de-lista

; Similar a la función 'member' de LISP
(defun miembro-de-lista (elemento lista)
    (cond ((null lista) nil)
          ((equal elemento (car lista)) lista)
          (t (miembro-de-lista elemento (cdr lista)))))

(print (miembro-de-lista 4 '(1 2 3 4 5 6)))
; RESULTADOS
; (4 5 6)

(print (miembro-de-lista 8 '(a b c d e f)))
; RESULTADOS
; NIL (8 no pertenece a la lista)
```

La función, empieza analizando si la lista que ha recibido es vacía, devolviendo NIL si lo está, en caso contrario entra a evaluar si la cabeza de la lista es igual al elemento a encontrar, cuando no es igual ella recursivamente vuelve a llamarse, pero ahora con la cola de la lista, comparando la cabeza de la lista hasta encontrar la que sea igual. Aquí los resultados del código anterior.

```
(4 5 6)
NIL
```

B. Función longitud lista

Una característica de las listas que resulta muy útil, es la función length la cual nos permite saber la longitud de la lista. Esto también se puede saber recursivamente, para ello se implementa el siguiente código:

```
; Función que calcula la longitud de la lista

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

 (print (longitud '(a b c d e f g h)))
; RESULTADOS
; 8
```

La función longitud recibe como parámetro la lista de la cual queremos saber su longitud, entrando a evaluar si la lista es vacía o no, devolviendo cero si lo es, de lo contrario realiza una suma por cada vez que se llame la función recursivamente, teniendo en cuenta que cada que se llama está enviando como parámetro la cola de la lista, sumando su cabeza hasta llegar al final.
A continuación, el resultado del código anteriormente propuesto:

```
8
```

A. Función termino n

LISP dispone de una función para extraer el enésimo elemento de una lista, denominado nth. Esta función también la podemos describir recursivamente como se demuestra a continuación:

```
; Función que extrae el enésimo término de la lista

(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)))))

 (print (termino-n 5 '(a b c d e f g h i j k l)))
; RESULTADOS
; F
```

La función termino-n recibe como parámetros el enésimo elemento y la lista a extraer el elemento.
Empieza evaluando si el enésimo elemento enviado es cero, devolviendo la cabeza de la lista cuando introducimos cero. Si dicho n no es cero, ella se vuelve a llamar recursivamente restándole 1 al enésimo n introducido y con la cola de la lista, esto para poder reducir el enésimo n a cero y así tener el objeto de la lista en la cabeza. El resultado del código anteriormente mencionado es el siguiente:

```
F
```

REFERENCIAS

Referencias en la Web:

[1] http://www.redesep.com/materias/blanda/5_2_estructuras_recursivas.ph