Estudio de algoritmos básicos en Lisp

Basics algorithms study in Lisp.

Javier Mauricio Gil Cardona
Moisés Alfonso Ducuara Ospina
Ingeniería de sistemas y computación, Universidad Tecnológica de Pereira, Pereira, Colombia
Correo-e1: mauriciogi@utp.edu.co
Correo-e2: moises.ducuara@utp.edu.co

***Resumen*— En este paper se mostrarán distintas funciones y sentencias escritas en el lenguaje Lisp y más concretamente en el listener de el software LispWorks. Con el fin de afianzar conocimientos en este lenguaje. Siendo este trabajo una entrada a la Inteligencia artificial.**

Palabras clave*— *Lenguaje Lisp, funciones, sentencias, Listener de compilador.

***Abstract*— This document shows different functions and statements written in the Lisp language and more specifically in the LispWorks software listener.In order to consolidate knowledge in this language. This work being an entrance to artificial intelligence.**

***Key Word* — Lisp language, functions, statements, compiler listener.**

I. INTRODUCCIÓN

Este documento es una recopilación de sentencias y funciones realizadas en el lenguaje Lisp para ser entregadas como trabajo correspondiente al curso de inteligencia artificial de la Universidad Tecnológica de Pereira el cual servirá para complementar lo aprendido en el lenguaje y en el software de LispWorks.

Para esto usaremos una de las herramientas que nos ofrece este software llamada listener, donde podremos poner nuestras sentencias y funciones y este las compila para mostrar un resultado a dicha función o sentencia.

II. CONTENIDO

La metodología será simple se iniciará con sentencias sentencias aritméticas sencillas y se irá subiendo poco a la complejidad de las funciones hasta llegar a las más robustas pondremos la sentencia o función seguido de su resultado.

En es este **TALLER 1** simplemente evaluaremos líneas de sentencias aritméticas.

```
CL-USER 1 > (* (/ (+ 3 3) (- 4 4)) (* 0 9))

Error: Division-by-zero caused by / of (6 0).
  1 (continue) Return a value to use.
  2 Supply new arguments to use.
  3 (abort) Return to level 0.
  4 Return to top loop level 0.

Type :b for backtrace or :c <option number> to proceed.
Type :bug-form "<subject>" for a bug report template or :? for other options.
```

FIGURA 1.1 Sentencia aritmética errónea

En la **FIGURA 1.1** si seguimos detenidamente la sentencias nos podemos dar cuenta que existe un error aritmético el cual es que en un punto divide por 0.

```
CL-USER 2 : 1 > (+ (* 3 (- 5 3)) (/ 8 4))
8
```

FIGURA 1.2 Sentencia aritmética exitosa.

En la **FIGURA 1.2** tenemos que el resultado es 8 y si la observamos detalladamente podemos ver que es correcta esa afirmación

```
CL-USER 3 : 1 > (* 3 2 2 (- 8 5))
36
```

FIGURA 1.3 Sentencia aritmética exitosa

En la **FIGURA 1.3** tenemos que el resultado es 36 y si la observamos detalladamente podemos ver que es correcta esa afirmación

Fecha de Recepción: xx/xx/xxxx
Fecha de Aceptación: Dejar en blanco

```
CL-USER 5 : 1 > ( (*) (* 2 (- 4 2)) (/ 8 2))

Error: Illegal argument in functor position: (*) in ((*) (* 2 (- 4 2)) (/ 8 2)).
  1 (continue) Evaluate (*) and ignore the rest.
  2 (abort) Return to level 1.
  3 Return to debug level 1.
  4 Return to level 0.
  5 Return to top loop level 0.

Type :b for backtrace or :c <option number> to proceed.
Type :bug-form "<subject>" for a bug report template or :? for other options.
```

FIGURA 1.4 sentencia aritmética con error de gramática de lenguaje

En la **FIGURA 1.4** vemos como LispWorks nos muestra un error el cual nos dice que la sentencia (*) no es correcta debido a que es una multiplicación vacía.

Las siguientes expresiones son un poco más avanzadas aunque sin salirnos de lo básico tanto en lo aritmético como en lo perteneciente al lenguaje de lisp, Se usará algunas palabras reservadas del lenguaje como lo son:

- sqrt: Raíz Cuadrada
- log: Logaritmo natural
- abs: Valor absoluto
- expt: Elevar a la potencia
- mod: Módulo de la división

```
CL-USER 6 : 2 > (sqrt (abs (- 71 (expt (+ 2 4) 4 ))))
35.0
```

FIGURA 1.5 Expresión aritmética con palabras reservadas

En la **FIGURA 1.5** tenemos que el resultado es 35.0 y si realizamos las operaciones manualmente, nos damos cuenta que el resultado que da la sentencia es correcta.

```
CL-USER 7 : 2 > '(' + '1 '(* 2 4))
((QUOTE +) (QUOTE 1) (QUOTE (* 2 4)))
```

FIGURA 1.6 Expresión aritmética con comilla o citas

En la **FIGURA 1.6** se puede observar que se están citando las listas es decir que lo que hay en su interior no será evaluado.

```
CL-USER 8 : 2 > (expt (mod 5 3) (abs (- 8 9)))
2
```

FIGURA 1.7 Expresión aritmética con palabras reservadas

En la **FIGURA 1.7** tenemos que el resultado es 2 y si realizamos las operaciones manualmente, nos damos cuenta que el resultado que da la sentencia es correcta.

```
CL-USER 9 : 2 > (quote (nil 'nil T 'T))
(NIL (QUOTE NIL) T (QUOTE T))
```

FIGURA 1.8 Expresión con listas citadas.

En la **FIGURA 1.8** podemos observar como el símbolo “ ‘ “ equivale a la palabra reservada del lenguaje quote, siendo ambos para hacer citados evitando la evaluación de las listas.

```
CL-USER 11 : 3 > (log (expt 2 (- 15 5 4)))
4.158883
```

FIGURA 1.9 Expresión aritmética con palabras reservadas.

En la **FIGURA 1.9** podemos apreciar cómo se ejecuta una sentencia que tiene operaciones aritmética y las ejecuta de tal forma que logra obtener el resultado correcto. Usando logaritmos y evaluando potencias.

```
CL-USER 1 : 1 >(quote (quote (Hello ByeBye)))
(QUOTE (HELLO BYEBYE))
```

FIGURA 1.10 Expresión con listas citadas.

El siguiente a realizar es el **TALLER 2** que corresponde a algunas asignaciones de variables.

```
CL-USER 2 : 2 > (setq A 4 B 6 C 5 X (+ A B) Y (- B C) Z (MAX A C))
5
```

FIGURA 2.1 Asignación de variables para realizar operaciones algebraicas.

En la **FIGURA 2.1** se puede observar cómo se asignan varias variables como por ejemplo que A = 4, B = 6, C = 5, y a su vez usar estas variables para su uso en álgebra como se observa en el apartado X (+ A B) que nos dice que X = A + B. La asignación de variables se convierte en algo más dinámico y funcional, al trabajar con múltiples operaciones y variables. En este lenguaje todo estos procesos mencionados se pueden representar en unas pocas líneas.

Para los siguientes ejemplos se usará la misma asignación de variables de la **FIGURA 2.1** pero con distintas evaluaciones.

```
CL-USER 3 : 2 > (+ A B C)
15
```

FIGURA 2.2 Expresión algebraica.

Si miramos detalladamente la **FIGURA 2.2** y las asignaciones de variable que se encuentran en la **FIGURA 2.1** y realizamos manualmente la operación veremos que el resultado es correcto.

```
CL-USER 2 > (eval x)
10
```

FIGURA 2.3 Evaluación de la variable X.

En la **FIGURA 2.3** simplemente vemos cómo se evalúa la variable X que se encuentra en el **FIGURA 2.1** la cual corresponde a la suma de A y B (4+6).

```
CL-USER 3 > (eval y)
1
```

FIGURA 2.4 Evaluación de la variable Y.

En la **FIGURA 2.4** simplemente vemos cómo se evalúa la variable Y que se encuentra en el **FIGURA 2.1** la cual corresponde a la resta de B y C (6-5).

```
CL-USER 6 : 3 > (eval z)
5
```

FIGURA 2.5 Evaluación de la variable Z.

En la **FIGURA 2.4** simplemente vemos cómo se evalúa la variable Z que se encuentra en el **FIGURA 2.1** la cual corresponde al mostrar el máximo valor entre A y C (4<5).

En el **TALLER 3** seguiremos haciendo asignaciones a variables esta vez las aplicaremos a otras operaciones matemáticas.

En la primera parte de este **TALLER 3** usaremos la misma asignación de variables que realizamos en el **TALLER 2** más exactamente en la **FIGURA 2.1**.

```
CL-USER 7 : 3 > (+ (* c z) b z)
36
```

FIGURA 3.1 Expresión algebraica con variables asignadas.

En la **FIGURA 3.1** se realiza una operación algebraica con las variables que se encuentran en la asignación de la **FIGURA 2.1.**

```
CL-USER 7 : 4 > (abs (+ (* (- z x a) -100) b))
906
```

FIGURA 3.2 Expresión algebraica con variables asignadas.

En la **FIGURA 3.1** se realiza una operación algebraica con las variables que se encuentran en la asignación de la **FIGURA 2.1.**

```
CL-USER 2 > (* (+ (* c x) (- a z c)) 2 y)
88
```

FIGURA 3.3 Expresión algebraica con variables asignadas.

En la **FIGURA 3.3** se realiza una operación algebraica con las variables que se encuentran en la asignación de la **FIGURA 2.1.**

```
CL-USER 3 > (setq m (+ z a) n (- y c) p (* 2 z))
10
```

FIGURA 3.4 Asignación de variables y operaciones algebraicas.

En la **FIGURA 3.4** podemos observar que se hace una nueva asignación de variables las cuales están conformadas sobre las variables que teníamos anteriormente las cuales se muestran en la **FIGURA 2.1.**

```
CL-USER 4 > (+ M Z X Y P B N)
37
```

FIGURA 3.5 Expresión algebraica con variables asignadas.

En la **FIGURA 3.5** podemos observar una suma que está conformada por variables que se encuentran en la **FIGURA 2.1** y en la **FIGURA 3.4**.

```
CL-USER 5 > (- (- (* 3 Z) (/ 100 C)) A C N P)
-20
```

FIGURA 3.6 Expresión algebraica con variables asignadas.

En la **FIGURA 3.5** podemos observar una expresión algebraica que está conformada por variables que se encuentran en la **FIGURA 2.1** y en la **FIGURA 3.4**.

```
CL-USER 1 > (cons (car '(axl wil rich)) (cdr '(este anto alla)))
(AXL ANTO ALLA)
```

FIGURA 4.1 Expresión que separa componentes de una lista.

En la **FIGURA 4.1** es una especie de juego sobre el uso de las "cabezas" y las "colas" de las listas, este componente de lisp es uno de los más utilizados. Es una herramienta de gran peso para el lenguaje.

```
CL-USER 3 : 1 > (cons (cdar '((cons Go Up))) (third '(we find (all fine))))
((GO UP) ALL FINE)
```

FIGURA 4.2 Expresión que crea lista nueva.

En la **FIGURA 4.2** se puede apreciar cómo se seleccionan conscientemente los elementos de una lista, para generar una nueva lista diferente de la lista donde fueron creados.

```
CL-USER 5 : 2 > (CAR (CDR (CAR (CDR '((a b) (c d) (e f))))))
D
```

FIGURA 4.3 Expresión que entre listas libera un elemento.

En la **FIGURA 4.3** se muestra un manejo de elementos que separan partes de una lista, en busca de un elemento específico de 3 listas diferentes, cada una con dos elementos en ellas.

```
CL-USER 6 : 2 > (car (car (cdr (cdr '((a b) (c d) (e f))))))
E
```

FIGURA 4.4 Expresión que entre listas libera un elemento.

En la **FIGURA 4.4** se muestra un manejo de elementos que separan partes de una lista, en busca de un elemento específico de 3 listas diferentes, cada una con dos elementos en ellas.

```
CL-USER 7 : 2 > (car (car (cdr '(cdr ((a b) (c d) (e f))))))
(A B)
```

FIGURA 4.5 Expresión que entre listas libera un elemento.

En la **FIGURA 4.5** se muestra un manejo de elementos que separan partes de una lista, en busca de una lista específica de 3 listas diferentes, cada una con dos elementos en ellas.

```
CL-USER 8 : 2 > '(car (car (cdr (cdr ((a b) (c d) (e f))))))
(CAR (CAR (CDR (CDR #))))
```

FIGURA 4.6 Expresión que no procesa una lista de listas.

En la **FIGURA 4.6** se muestra un manejo de elementos, pero en este caso no se ejecuta ninguno, ya que la comilla del principio evita que que la lista se ejecute.

```
CL-USER 9 : 2 > (cons (car nil) (cdr nil))
(NIL)
```

FIGURA 4.7 Expresión que concluye con el vacío.

En la **FIGURA 4.7** se puede apreciar cuál es la cabeza y la cola en una lista vacía. El resultado en ambos es vacío.

```
CL-USER 10 : 2 > (cdr (car (cdr (car '((D (E F)) G (H I))))))
(F)
```

FIGURA 4.8 Expresión que entre listas libera un elemento.

En la **FIGURA 4.8** se muestra un manejo de elementos que separan partes de una lista, en busca de un elemento específico. En este caso la lista es más diversa.

```
CL-USER 12 : 3 > (setq a '(+ 3 6))
(+ 3 6)
```

FIGURA 4.9.1 Expresión aritmética que es asignada a una variable.

En la **FIGURA 4.9.1** se muestra cómo se asigna una variable, pero en este caso no es un valor numérico, sino una operación aritmética sin resolver. La comilla que lo antecede genera este comportamiento.

```
CL-USER 2 > (CDR A)
(3 6)
```

FIGURA 4.9.2 Expresión retorna los dos últimos elementos.

En la **FIGURA 4.9.2** se muestra que la cola de la lista, no es un elemento concreto, ni valor nulo. Si no valores numéricos que de la suma que no se ejecutó en la variable.

```
CL-USER 6 : 1 > (CAR (CDR A))
3
```

FIGURA 4.9.3 Expresion que excluye un elemento de la lista.

En la **FIGURA 4.9.3** se muestra una pequeña búsqueda de un elemento que está en la variable A. Esta búsqueda se ejecuta usando los elementos que dividen las listas ("CAR" y "CDR").

```
CL-USER 3 > (CAR (CDR (CDR A)))
6
```

FIGURA 4.9.4 Expresion que excluye un elemento de la lista.

En la **FIGURA 4.9.4** se muestra una pequeña búsqueda de un elemento que está en la variable A. Esta búsqueda se ejecuta usando los elementos que dividen las listas (“CAR” y “CDR”).

En el **TALLER 5** realizaremos unas combinaciones de CAR y CDR para que el programa nos muestre el símbolo mapache, el cual se encuentra en distintas combinaciones de lista.

```
CL-USER 2 : 1 > (car (cdr (cdr '(oso gato mapache ardilla))))
MAPACHE
```

FIGURA 5.1 Expresión que busca “mapache” en la lista.

En la **FIGURA 5.1** se procede a buscar el arreglo de caracteres “mapache” con el uso de unos elementos de división de listas, estos fueron: Una cabeza y dos colas.

```
CL-USER 3 : 1 > (car (car (cdr '((oso gato) (mapache ardilla)))))
MAPACHE
```

FIGURA 5.2 Expresión que busca “mapache” en la lista.

En la **FIGURA 5.2** se procede a buscar el arreglo de caracteres “mapache” con el uso de unos elementos de división de listas, estos fueron: Dos cabezas y un cola.

```
CL-USER 16 : 4 > (car (car (cdr (cdr (car '(((oso) (gato) (mapache) (ardilla)))))))
MAPACHE
```

FIGURA 5.3 Expresión que busca “mapache” en la lista.

En la **FIGURA 5.3** se procede a buscar el arreglo de caracteres “mapache” con el uso de unos elementos de división de listas, estos fueron: Tres cabezas y dos colas.

```
CL-USER 19 : 4 > (car (car (car (cdr (cdr '(oso (gato) ((mapache)) (((ardilla )))))))))
MAPACHE
```

FIGURA 5.4 Expresión que busca “mapache” en la lista.

En la **FIGURA 5.4** se procede a buscar el arreglo de caracteres “mapache” con el uso de unos elementos de división de listas, estos fueron: Tres cabezas y dos colas.

En el **TALLER 6** realizaremos algo muy similar a lo que hicimos en el **TALLER 5** pero esta vez mostraremos símbolos distintos que se encuentran en otro orden, los símbolos serán León y Fútbol. Otra cosa a tener en cuenta es que esta vez no tenemos que usar únicamente CDR y CAR si no que también podremos usar otras funciones y palabras reservadas propias del lenguaje Lisp.

```
CL-USER 21 : 5 > (third (cdr '(Managua Chinandega Rivas León Boaco)))
LEÓN
```

FIGURA 6.1 Expresión que busca “león” en la lista.

En la **FIGURA 6.1** se procede a buscar el arreglo de caracteres “león” con el uso de unos elementos de división de listas, estos fueron: Un tercer y una cola.

```
CL-USER 22 : 5 > (third (second '((managua) (Chinandega Rivas León) Boaco)))
LEÓN
```

FIGURA 6.2 Expresión que busca “león” en la lista.

En la **FIGURA 6.2** se procede a buscar el arreglo de caracteres “león” con el uso de unos elementos de división de listas, estos fueron: Un tercer y una segundo.

```
CL-USER 4 > (second (second (second '(Managua (Chinandega (Rivas León Boaco))))))
LEÓN
```

FIGURA 6.3 Expresión que busca “león” en la lista.

En la **FIGURA 6.3** se procede a buscar el arreglo de caracteres “león” con el uso de unos elementos de división de listas, estos fueron: Tres segundos.

```
CL-USER 20 : 3 > (car (car (third '(Deportes (Béisbol tenis) ((fútbol) billar)))))
FÚTBOL
```

FIGURA 6.4 Expresión que busca “fútbol” en la lista.

En la **FIGURA 6.4** se procede a buscar el arreglo de caracteres “fútbol” con el uso de unos elementos de división de listas, estos fueron: Dos cabezas y un tercero**.**

En el **TALLER 7** evaluaremos algunas expresiones con algunas funciones que ya hemos vistos y algunas otras nuevas como “append” que lo que hace es agregar un elemento nuevo a una lista.

```
CL-USER 22 : 4 > (cons (second '(leo mana china)) (cons (list (car '(1 2 3 4)) (cdr '(a b c d)))'2))
(MANA (1 (B C D)) . 2)
```

FIGURA 7.1 Expresión que genera una lista más rica.

La **FIGURA 7.1** corresponde simplemente a una evaluación de sentencia correspondiente al primer punto del **TALLER 7**.

```
CL-USER 23 : 4 > (LIST (LIST '(ca ce) (LAST '(0 a 1 b ci) (THIRD '(5 4 3 2 1))) (NTH 3 '(pa pe pi po
 pu))))
(((CA CE) (1 B CI) PO))
```

FIGURA 7.2 Expresión que genera una lista más rica.

La **FIGURA 7.2** corresponde simplemente a una evaluación de sentencia correspondiente al segundo punto del **TALLER 7**.

```
CL-USER 1 > (list (append '(h o l a) '(m u n d o)) (cddr '(sal pan arroz pollo)) (car '(buen mal)) (cadr '(desayuno recreo almuerzo cena)))
((H O L A M U N D O) (ARROZ POLLO) BUEN ALMUERZO)
```

FIGURA 7.3 Expresión que genera una lista más rica.

La **FIGURA 7.3** corresponde simplemente a una evaluación de sentencia correspondiente al tercer punto del **TALLER 7**.

```
CL-USER 3 : 1 > (list '(¿como estas?) (nth 0 '(bien mal rematado)) (append (car '((estas) estoy)) (cdr '(entiendo entendiendo lisp))))
((¿COMO ESTAS?) BIEN (ESTAS ENTENDIENDO LISP))
```

FIGURA 7.4 Expresión que genera una lista más rica.

La **FIGURA 7.4** corresponde simplemente a una evaluación de sentencia correspondiente al cuarto punto del **TALLER 7**.

```
CL-USER 4 : 1 > (length (cons (car '(verdad mentira falso)) (list* 'es 'muy '(facil))))
4
```

FIGURA 7.5 Expresión que genera una lista más rica.

La **FIGURA 7.5** corresponde simplemente a una evaluación de sentencia correspondiente al quinto punto del **TALLER 7**.

```
CL-USER 6 : 2 > (cdr (list (subseq '(z y x w v) 0 2) (cons '(a e i) '(o u))))
(((A E I) O U))
```

FIGURA 7.6 Expresión que genera una lista más rica.

La **FIGURA 7.6** corresponde simplemente a una evaluación de sentencia correspondiente al último punto del **TALLER 7**.

En el **TALLER 8** asignaremos una variable ‘países’ que contendrá una pequeña base de datos con un país y su respectiva capital.

```
CL-USER 7 : 2 > (setq paises '((Nicaragua . Managua) (Italia . Roma) (España . Madird)))
((NICARAGUA . MANAGUA) (ITALIA . ROMA) (ESPAÑA . MADIRD))
```

FIGURA 8.1 Base de datos de países

En la **FIGURA 8.1** construimos la base de datos mostrada en las diapositivas, la cual corresponde al **TALLER 8**.

```
CL-USER 8 : 2 > (list 'Nicaragua 'Italia 'España (sublis paises '(Nicaragua Italia España)) (nth 1 '(eran son serán)) (car (nthcdr 1 (cdr '(pueblos estados capitales barrios)))) (cons 'de (cons 'estos (cons 'paises nil))))
(NICARAGUA ITALIA ESPAÑA (MANAGUA ROMA MADIRD) SON CAPITALES (DE ESTOS PAISES))
```

FIGURA 8.2 Interactuar con la base de datos

En la **FIGURA 8.2** agregamos algunas explicaciones en texto a el contenido de la base de datos que se encuentra en la **FIGURA 8.1**

En el **TALLER 9** haremos algunas bases de datos las cuales corresponde a palabras con sus respectivos sinónimos, antónimos y traducción a inglés, posterior a esto uniremos las palabras con sus respectivos sinónimos, antónimos y traducción al inglés. Para esto usaremos la palabra reservada pairlis que lo que hace es unir en una lista las posiciones correspondientes.

```
CL-USER 11 : 4 > (setq Palabras '(grande bonito feliz humedo) Sinonimos '(alto bello contento mojado) antonimos '(pequeño feo triste seco) inglés '(big beauty happy humid))
(BIG BEAUTY HAPPY HUMID)
```

FIGURA 9.1 Bases de datos con palabras, sinónimos, antónimos y su traducción en inglés.

Creamos bases de datos con distintos contenidos relacionados entre sí.

```
CL-USER 12 : 4 > (setq palabras-sinonimos (pairlis palabras sinonimos))
((HUMEDO . MOJADO) (FELIZ . CONTENTO) (BONITO . BELLO) (GRANDE . ALTO))
```

FIGURA 9.2 Sentencia con pairlis aplicada a dos listas.

En La **FIGURA 9.2** juntamos palabras con su respectivo sinónimo usando la palabra reservada pairlis.

```
CL-USER 14 : 5 > (setq palabras-antonimos (pairlis palabras antonimos))
((HUMEDO . SECO) (FELIZ . TRISTE) (BONITO . FEO) (GRANDE . PEQUEÑO))
```

FIGURA 9.3 Sentencia con pairlis aplicada a dos listas.

En La **FIGURA 9.3** juntamos palabras con su respectivo antónimo usando la palabra reservada pairlis.

```
CL-USER 15 : 5 > (setq palabras-inglés (pairlis palabras inglés))
((HUMEDO . HUMID) (FELIZ . HAPPY) (BONITO . BEAUTY) (GRANDE . BIG))
```

FIGURA 9.4 Sentencia con pairlis aplicada a dos listas.

En La **FIGURA 9.4** juntamos palabras con su respectiva traducción al inglés usando la palabra reservada pairlis.

En el **TALLER 10** usamos la base de datos que armamos en el **TALLER 9** más exactamente en la **FIGURA 9.1.** Lo que hacemos es asociar algunos símbolos de listas correspondientes a la **FIGURA 9.2, FIGURA 9.3 y FIGURA 9.4.**

```
CL-USER 7 : 2 > (cdr (assoc 'grande Palabras-Inglés))
BIG
```

FIGURA 10.1 Asociación de Palabras a una lista.

En la **FIGURA 10.1** le buscamos al símbolo “Grande” su respectiva traducción al Inglés la cual corresponde a “Big”

```
CL-USER 9 : 3 > (cdr (assoc 'feliz Palabras-Sinonimos))
CONTENTO
```

FIGURA 10.2 Asociación de Palabras a una lista.

En la **FIGURA 10.2** le buscamos al símbolo “Feliz” su respectivo sinónimo el cual corresponde a “Contento”

```
CL-USER 11 : 3 > (first (assoc 'húmedo (acons 'pal 'anto Palabras-Antónimos)))
HÚMEDO
```

FIGURA 10.3 Asociación de Palabras a una lista.

En la **FIGURA 10.3** le buscamos al símbolo “Húmedo” el antónimo del antónimo el cual corresponde al símbolo “Húmedo” nuevamente

III. CONCLUSIONES

Este taller nos proporciona un conocimiento un poco más amplio del uso de la herramienta LispWorks. Cada una de las sentencia, nos hace dudar de cómo puede llegar a funcionar, afianzando elementos ya conocidos y descubriendo otras que hasta el momento no se tomaban en cuenta. En general, como está planteado de manera escalonada, cada elemento pudiera volverse cada vez más complejo, logra un aprendizaje mucho más sencillo para el estudiante. Se buscan pequeñas metas a corto plazo, pero que a la larga puede trazar un gran camino.
El método de tener los componente listos para probarse da la apertura de preguntarse ciertas cosas y experimentar con los elementos existentes, descubriendo una pequeña creatividad al ser libre y fácil de implementar, sin tener consecuencias de pérdida de tiempo o de ver afectada la nota.

IV. REFERENCIAS

Referencias digitales:

[1] Clase virtual dada por el docente 15/04/2020

[2] Taller propuesto por el docente 15/04/2020

[3] http://www.redesep.com/materias/inteligencia/index.php