



¿Destruktivo?

En *Common LISP*, una de las características más importantes de una función es si opera de forma destruktiva o no...

```
>> (setq lista '(a b c 1 2 3))
(A B C 1 2 3)
>> (rest lista)
(B C 1 2 3)
>> lista
(A B C 1 2 3) ↓
>> (setq lista (rest lista))
(B C 1 2 3)
>> lista
(B C 1 2 3) ↑
```

Dr. Salvador Godoy Calderón

Parejas...

Muchas funciones existen en parejas, una versión destruktiva y la otra no-destruktiva...

<i>no-destructiva</i>	<i>destructiva</i>	<i>operación</i>
reverse	nreverse	Invertir una lista
append	nconc	Concatenar dos listas
remove	delete	Eliminar elementos
cl-subst	cl-nsubst	Reemplazar elementos
-	sort	Ordenar una lista
-	cl-merge	Fusionar dos listas

Dr. Salvador Godoy Calderón

Modificación universal...

La función **SETF** permite modificar destrutivamente cualquier *posición* en cualquier estructura de datos...

```
(setf <posición> <valor>)
```

```
>> (defparameter *lista* '(a b c d))  
*LISTA*  
>> (setf (third *lista*) 'x)  
X  
>> *lista*  
(A B X D)
```

Dr. Salvador Godoy Calderón

Listas como pilas y conjuntos...

Pilas...

Common Lisp incluye funciones para manejo de listas que representan una pila...

PUSH y **POP**
funcionan de forma
DESTRUCTIVA...

```
>> (defparameter *pila* '())  
*PILA*  
>> (push 'B *pila*)  
(B)  
>> (push 'A *pila*)  
(A B)  
>> (pop *pila*)  
A  
>> *pila*  
(B)
```

Dr. Salvador Godoy Calderón

Conjuntos...

También existen funciones para manejo de listas que representan conjuntos...

adjoin
member
union
intersection
set-difference
subsetp

agregar elemento a conjunto...
encontrar elemento en conjunto...
unión de conjuntos...
intersección de conjuntos...
diferencia entre conjuntos...
identificador de subconjuntos...

Dr. Salvador Godoy Calderón

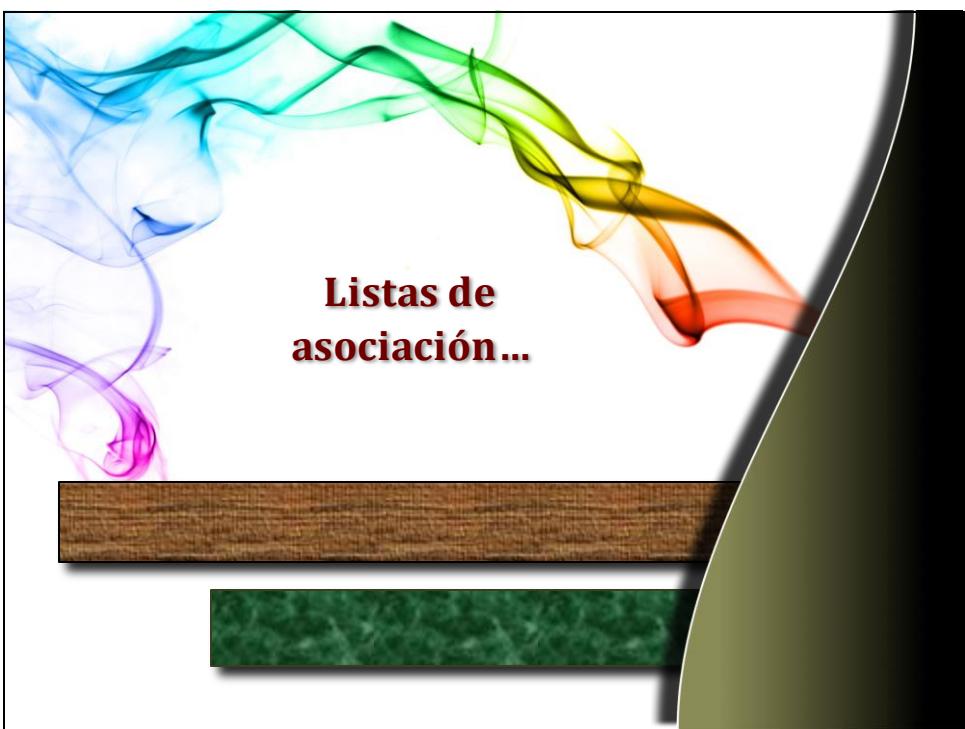
Ejemplos...

```
>> (setq A '(a b c d))
(A B C D)
>> (adjoin 'e A)
(E A B C D)
>> A
(A B C D)
>> (setq C (adjoin 'z A))
(Z A B C D)
>> C
(Z A B C D)
>> (adjoin 'c A)
(A B C D)
```

!Ya existe!

```
>> (setq B '(1 2 3 4))
(1 2 3 4)
>> (union A B)
(D C B A 1 2 3 4)
>> (union B A)
(4 3 2 1 A B C D)
>> (intersection (union B A) A)
(D C B A)
>> (intersection (union A B) A)
(A B C D)
>> (subsetp '(c d a) A)
T
```

Dr. Salvador Godoy Calderón



Listas de asociación...

¿Qué es una lista de asociación?...

Una lista de asociación (*alist*) es una lista que contiene parejas llave–valor. Cada una de esas parejas es una asociación

```
( (uno . 1) (dos . 2) (tres . 3) (cuatro . 4) ... )
```

Es una asociación entre dos conjuntos distintos, apareando o emparejando los elementos en la misma posición...

$$\begin{matrix} Llaves &= \{un, do, tre, cuatro\} \\ Valores &= \{1, 2, 3, 4\} \end{matrix}$$

Dr. Salvador Godoy Calderón

Creación...

Para crear listas de asociación se usa **pairlis**...

```
( pairlis <lista1> <lista2> [<lista-asociación>] )
```

```
>> (setq A '(uno dos tres cuatro))
(UNO DOS TRES CUATRO)

>> (setq B '(1 2 3 4))
(1 2 3 4)

>> (pairlis A B)
((CUATRO . 4) (TRES . 3) (DOS . 2) (UNO . 1))
```

Dr. Salvador Godoy Calderón

El tercer argumento...

Si a **pairlis** se le proporciona el tercer argumento (una lista de asociación), entonces agrega las parejas formadas a la lista indicada (de forma no-destructiva):

```
>> (setq Lista-A '((diez . 10) (once . 11)) )
( (DIEZ . 10) (ONCE . 11))

>> (pairlis A B Lista-A )
( (CUATRO . 4) (TRES . 3) (DOS . 2) (UNO . 1) (DIEZ . 10) (ONCE . 11) )

>> Lista-A
( (DIEZ . 10) (ONCE . 11))
```

Dr. Salvador Godoy Calderón

Agregar...

También se puede usar la función **acons** para agregar una asociación a una lista de asociaciones (también de forma no-destructiva):

(**acons** <llave> <valor> <lista-asociación>)

```
>> (acons 'nueve 9 Lista-A )
( (NUEVE . 9) (DIEZ . 10) (ONCE . 11))

>> Lista-A
'( (DIEZ . 10) (ONCE . 11))
```

Dr. Salvador Godoy Calderón

Buscar...

Para recuperar la información contenida en una lista de asociación existen cuatro funciones:

assoc , assoc-if , rassoc , rassoc-if

```
>> (setq lista '((x . 100) (y . 200) (z . 50)))
((X . 100) (Y . 200) (Z . 50))
>> (assoc 'y lista)
(Y . 200)
>> (assoc 'x lista)
(X . 100)
>> (assoc 'w lista)
NIL
```

Dr. Salvador Godoy Calderón

Assoc-If...

La función **assoc-if** opera de forma muy semejante a **assoc** pero entrega la primera asociación que cumple alguna condición expresada sobre su llave:

```
>> (setq lista2 '((1 . uno) (2 . dos) (3 . tres) (4 . cuatro)))
((1 . UNO) (2 . DOS) (3 . TRES) (4 . CUATRO))
>> (assoc-if #'evenp lista2)
(2 . DOS)
>> (assoc-if #'oddp lista2)
(1 . UNO)
```

Dr. Salvador Godoy Calderón

Al revés...

rassoc y **rassoc-if** rastrean la lista de asociaciones en el sentido opuesto de la relación, es decir, buscando la asociación a partir de algún valor:

```
>> (setq lista3 '((uno . 1) (dos . 2) (tres . 3)))
((UNO . 1) (DOS . 2) (TRES . 3))

>> (rassoc 2 lista3)
(DOS . 2)

>> (rassoc-if #'evenp lista3)
(DOS . 2)
```

Dr. Salvador Godoy Calderón

Nota...

Si la lista de asociaciones contiene listas propias, entonces **assoc** también funciona , sin embargo, **rassoc** no:

```
>> (setq otralista '((uno 1) (dos 2) (tres 3)))
((UNO 1) (DOS 2) (TRES 3))

>> (assoc 'dos otralista)
(DOS 2)

>> (rassoc 2 otralista)
NIL
```

Dr. Salvador Godoy Calderón



Registros

Registros...

Al igual que la mayoría de los lenguajes de programación *Common Lisp* permite la creación de estructuras de registro:

```
( defstruct <etiqueta> <campo1> <campo2> ...)
```

Opcionalmente se puede colocar un valor inicial por omisión para cada campo.

```
( defstruct persona nombre paterno materno )
```

Dr. Salvador Godoy Calderón

Lo importante...

La cualidad única de *Common Lisp* es que, al definir una estructura de registro, automáticamente define también funciones para su manejo:

```
make-<etiqueta>
<etiqueta>-p
<etiqueta>-<campo#1>
<etiqueta>-<campo#2>
...
<etiqueta>-<campo#n>
```

constructor de registros
predicado identificador del tipo
funciones de acceso a cada campo...

Dr. Salvador Godoy Calderón

Ejemplo...

```
>>(defstruct persona nombre paterno materno)
PERSONA
```

Los campos también adoptan valor por omisión **NIL**

```
>>(defparameter Yo (make-persona :nombre 'Salvador
                               :paterno 'Godoy
                               :materno 'Calderón) )
```

```
#S (PERSONA :NOMBRE SALVADOR :PATERNO GODOY :MATERNO CALDERÓN)
```

```
>>(Persona-p Yo)
T
```

```
>> (Persona-paterno Yo)
GODOY
>> (Persona-materno Yo)
CALDERÓN
>> (Persona-nombre Yo)
SALVADOR
```

Dr. Salvador Godoy Calderón

Jerarquías...

Es posible definir estructuras jerarquizadas (por inclusión) para darle forma a algún dominio de conocimiento:

```
>> (defstruct Persona nombre paterno materno )
PERSONA
```

```
>> (defstruct (Astronauta (:include Persona))
              tamaño_casco
              (bebida_favorita 'Tang) )
ASTRONAUTA
```

Astronauta incluye los mismos campos que **persona**, pero además, las mismas funciones definidas para **persona**...

Dr. Salvador Godoy Calderón

Entidades...

```
>> (setq X1 (make-Astronauta :nombre 'Yuri
                               :paterno 'Gagarin
                               :tamaño_casco 17.5
                               :bebida_favorita 'Vodka) )
#S(ASTRONAUTA :NOMBRE YURI :PATERNO GAGARIN :MATERNO NIL ...)
```

```
>> (setq X2 (make-Astronauta :nombre 'Neil
                               :paterno 'Alden
                               :materno 'Armstrong
                               :tamaño_casco 16) )
#S(ASTRONAUTA :NOMBRE NEIL :PATERNO ALDEN ...)
```

Dr. Salvador Godoy Calderón

Funciones...

```
>> (Astronauta-p X1)
T
>> (Persona-p X2 )
T
>> (Astronauta-bebida_favorita X2 )
TANG
>> (Astronauta-bebida_favorita X1 )
VODKA
>> (Astronauta-materno X1 )
NIL
>> (Persona-materno X2 )
ARMSTORNG
```

Dr. Salvador Godoy Calderón

Arreglos



Tipos y definición...

Common Lisp soporta arreglos multi-dimensionales como la mayoría de los lenguajes de programación...

Para crear un arreglo se usa **make-array** y se indica el número de dimensiones y la capacidad de almacenamiento en cada dimensión...

```
>> (make-array '(5) )
#(NIL NIL NIL NIL NIL)

>> (make-array '(3 4) )
#2A( (NIL NIL NIL NIL) (NIL NIL NIL NIL) (NIL NIL NIL NIL))

>> (make-array '(2 2 2) )
#3A( ((NIL NIL) (NIL NIL)) ((NIL NIL) (NIL NIL)) )
```

Dr. Salvador Godoy Calderón

Opciones...

Se puede especificar un valor inicial para todas las celdas, o bien para cada una:

```
>> (make-array '(5) :initial-element "HOLA")
#("HOLA" "HOLA" "HOLA" "HOLA" "HOLA")
```

```
>> (make-array '(2 4) :initial-contents '((0 1 2 3) (3 2 1 0)) )
#2A((0 1 2 3) (3 2 1 0))
```

Dr. Salvador Godoy Calderón

Acceso...

Para tener acceso a los elementos de un arreglo se usan las funciones **aref** y **setf**...

aref (array reference) es un “posicionador” que señala alguna posición (un índice) dentro de un arreglo y por lo tanto, sirve para leer el contenido de dicha posición...

setf es una función general de escritura en el lenguaje que “escribe” algún valor en alguna localidad de memoria bien especificada...

Dr. Salvador Godoy Calderón

Acceso...

```
>> (defvar *arreglo* (make-array '(3 4)) )
*ARREGLO*

>> (aref *arreglo* 2 2)
NIL

>> (setf (aref *arreglo* 2 2) "Manzana")
"MANZANA"

>> *arreglo*
#2A((NIL NIL NIL NIL) (NIL NIL NIL NIL) (NIL NIL "Manzana" NIL))

>> (aref *arreglo* 2 2)
"Manzana"
```

Dr. Salvador Godoy Calderón

Sin restricción de tipos...

```
>> (defvar *A* (make-array 10)) )
*A*
>> (setf (aref *A* 3) "Hola")
"HOLA"
>> (setf (aref *A* 5) 34/69)
34/69
>> (setf (aref *A* 7) '(a b c))
(A B C)
>> (setf (aref *A* 9) 142.698)
142.698
>> *A*
#(NIL NIL NIL "HOLA" NIL 34/69 NIL (A B C) NIL 142.698)
```

Dr. Salvador Godoy Calderón

Arreglos dinámicos...

A un arreglo que se le puede modificar su tamaño (dimensiones) se le llama arreglo dinámico...

```
>> (setq A (make-array '(3 2) :adjustable T
                         :initial-contents '((A B) (C D) (E F)) )
#2A( (A B) (C D) (E F) )
>> (adjust-array A '(2 4) )
#2A( (A B NIL NIL) (C D NIL NIL) )
```

Si el nuevo tamaño es mayor se agregan elementos, si el nuevo tamaño es menor se eliminan elementos ...

Dr. Salvador Godoy Calderón

Otras opciones...

El número de dimensiones
del arreglo...

```
>> (array-rank A)  
2
```

```
>> (array-dimensions A)  
(2 4)
```

La capacidad en cada
dimensión de un arreglo...

El número máximo de
elementos en el arreglo...

```
>> (array-total-size A)  
8
```

Dr. Salvador Godoy Calderón

Tablas Hash



Tablas hash...

Existen en *LISP* estructuras de datos más eficientes que otras, aunque menos flexibles:

Listas → Arreglos

Listas de asociación → Tablas hash
(alists)

Una tabla hash es una estructura para almacenar parejas llave – valor en forma eficiente...

Su principal ventaja es que el tiempo de acceso a cada elemento es CONSTANTE ...

Dr. Salvador Godoy Calderón

Desventajas...

- ★ Con una tabla hash sólo se puede consultar la asociación de forma directa...
- ★ El compilador no puede fácilmente desplegar el contenido de la tabla...

```
>> (defparameter *Tabla* (make-hash-table))
*TABLA*
>> *Tabla*
#<HASH-TABLE :TEST EQL :COUNT 0 {1004522133}>
```

Dr. Salvador Godoy Calderón

Creación...

Para crear una tabla hash se usa **make-hash-table...**

```
(make-hash-table :test :size :rehash-size :rehash-threshold)
```

:test Función para comparar llaves: **#'eq #'eql #'equal**

:size Tamaño inicial (en elementos) de la tabla...

:rehash-size Número de elementos a agregar cuando se llena la tabla...

:rehash-threshold Nivel de llenado requerido para aumentar el tamaño de la tabla...

Dr. Salvador Godoy Calderón

Observación...

De manera normal, las tablas hash usan **eql** para comparar la llave de una búsqueda...

Si usamos una lista como llave de una tabla hash, resulta necesario indicarle que debe usar **equal** para manejo de las llaves:

```
>> (setq *Tabla* (make-hash-table :test #'equal))  
#S(HASH-TABLE ...)
```

Dr. Salvador Godoy Calderón

Uso...

make-hash-table se puede invocar sin argumentos...

```
>> (setq *Tabla* (make-hash-table) )  
#S(HASH-TABLE ...)
```

Para insertar y buscar un valor en la tabla se usa la función **gethash...**

```
>> (gethash </>av> <tabla>)  
<valor>  
T  
  
>> (setf (gethash </>av> <tabla>) <valor>)  
<valor>
```

Dr. Salvador Godoy Calderón

Uso...

make-hash-table se puede invocar sin argumentos...

```
>> (setq *Tabla* (make-hash-table) )  
#S(HASH-TABLE ...)
```

Para eliminar un valor un valor se usa **remhash...**

```
>> (remhash </>av> Tabla)  
T
```

Dr. Salvador Godoy Calderón

Ejemplo celdas y cadenas...

```
>> (defparameter *Tabla* (make-hash-table :test #'equal) )  
#S(HASH-TABLE ...)  
>> (setf (gethash '(A . 0) *Tabla*) "Uno" )  
"Uno"  
>> (setf (gethash '(B . 1) *Tabla*) "Dos" )  
"Dos"  
>> (setf (gethash '(C . 2) *Tabla*) "Tres" )  
"Tres"
```

Dr. Salvador Godoy Calderón

