

# Fundamentos de programación

## Datos complejos II: Funciones recursivas auxiliares, listas de listas y definiciones locales

[carlos.andres.delgado@correounivalle.edu.co](mailto:carlos.andres.delgado@correounivalle.edu.co)

Carlos Andrés Delgado S.

Facultad de Ingeniería. Universidad del Valle

Octubre de 2016

Fundamentos  
de  
programación

Carlos Andrés  
Delgado S.

Funciones  
recursivas  
auxiliares

Listas en listas

Abreviaciones  
de listas

Definiciones  
locales

**1** Funciones recursivas auxiliares

**2** Listas en listas

**3** Abreviaciones de listas

**4** Definiciones locales

Fundamentos  
de  
programación

Carlos Andrés  
Delgado S.

Funciones  
recursivas  
auxiliares

Listas en listas

Abreviaciones  
de listas

Definiciones  
locales

## 1 Funciones recursivas auxiliares

## 2 Listas en listas

## 3 Abreviaciones de listas

## 4 Definiciones locales

# Funciones recursivas auxiliares

Fundamentos  
de  
programación

Carlos Andrés  
Delgado S.

Funciones  
recursivas  
auxiliares

Listas en listas

Abreviaciones  
de listas

Definiciones  
locales

## Definición

El ordenamiento es un problema muy común en la computación. Por ejemplo, ordenar archivos por fecha o ordenar los correos electrónicos. Para el estudio de este problema nos basaremos en el ordenamiento de listas

# Funciones recursivas auxiliares

Fundamentos  
de  
programación

Carlos Andrés  
Delgado S.

Funciones  
recursivas  
auxiliares

Listas en listas

Abreviaciones  
de listas

Definiciones  
locales

## Definición

En el ordenamiento de números podemos observar:

```
;;Propósito: Crea una lista ordenada de números  
;;sort: lista-de-numeros -> lista-de-numeros  
(sort empty)  
->empty  
(sort (cons 2 (cons 4 (cons 1 empty))))  
->(cons 1 (cons 2 (cons 4 empty)))
```

# Funciones recursivas auxiliares

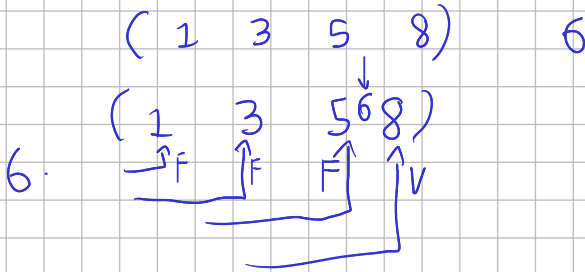
(5) 8 1 3)

## Definición

Una definición para la función que ordena números es:

```
;;Propósito: Crea una lista ordenada de números
;;sort: lista-de-numeros -> lista-de-numeros
(define (sort lista)
  (cond
    [(empty? lista) empty]
    [else ... ((first lista) ... [sort (rest lista))])
  ])
)
```

Función: Lista que está ordenada y un número



;Contrato: insertar: lista de numeros, numero -> lista de numeros

;;Ejempls (insertar (cons 1 (cons 2 (cons 4 empty))) 3)

;; (cons 1 (cons 2 (cons 3 (cons 4 empty))))

;;(insertar (cons 2 (cons 4 (cons 8 empty))) 12)

;;(cons 2 (cons 4 (cons 8 (cons 12 empty))))

(define (insertar lnum num)

(cond

[(empty? lnum) (cons num empty)]

[(< num (first lnum)) (cons num lnum)]

[else (cons (first lnum) (insertar (rest lnum) num))

)

)

## Definición

En la anterior función podemos ver:

- 1 (first lista) extrae el primer elemento de la lista
- 2 (sort (rest lista)) produce una versión ordenada del resto de la lista



## Definición

Sin embargo, esto no ordena la lista, ya que sólo permite explorar la lista de entrada

- 1 (first lista) extrae el primer elemento de la lista
- 2 (sort (rest lista)) produce una versión ordenada del resto de la lista

Por ello necesitamos una segunda función que nos permita crear la lista ordenada

## Definición

La función **insert** nos permite insertar un número en una lista ordenada de números

```
;;Propósito: Inserta un elementos en una lista ordenada  
;;insert: lista-de-numeros -> lista-de-numeros  
(define (insert n lista) ...)
```

# Funciones recursivas auxiliares

Fundamentos  
de  
programación

Carlos Andrés  
Delgado S.

Funciones  
recursivas  
auxiliares

Listas en listas

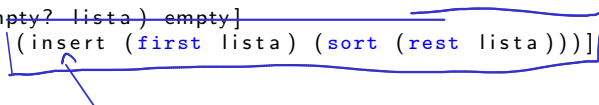
Abreviaciones  
de listas

Definiciones  
locales

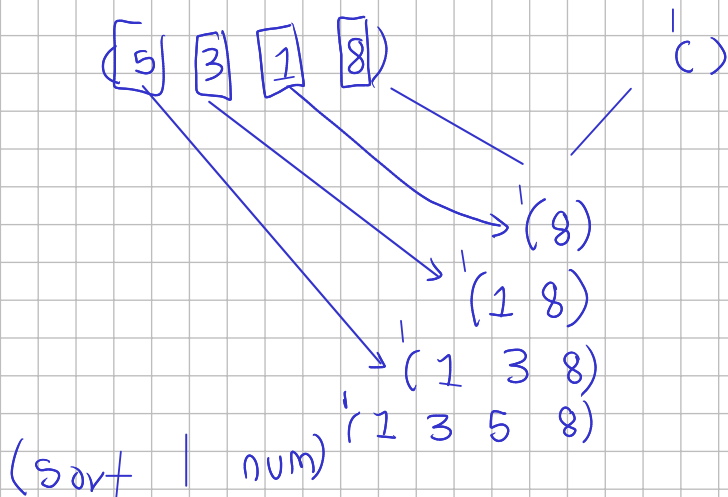
## Definición

Al incluir la función insertar podemos realizar el ordenamiento

```
;;Propósito: Crea una lista ordenada de números
;;sort: lista-de-numeros -> lista-de-numeros
(define (sort lista)
  (cond
    [(empty? lista) empty]
    [else (insert (first lista) (sort (rest lista)))]
  )
)
```



¿Pero como construimos la lista ordenada?



(sort (sort (sort '() 8) 1) 3)

'(8)  
'(1 8)  
'(1 8 3)

(sort (sort (sort (sort '() 8) 1) 3) 5)

(define (ordenar lnum)

(cond

[(empty? lnum) '()]

[else

(insertar (ordenar (rest lnum)) (first num)))]))

→ '(s 8 4 2 3 9)

(insertar

(insertar

'(2 3 4 8 9)

'(2 3 4 5 8 9)

(insertar ← '(2 3 4 9)

(insertar

(insertar

(insertar '())

'(9)

'(3 9)

'(2 3 9)

2) 3)

4) 8)

5.)

(9 7 5 1 3 6)

(list 1 3 5 6 7 9)

Ordenar

tratar la  
lista

Insertar



Ordenar

Insertar un número  
a una lista ordenada

# Funciones recursivas auxiliares

Fundamentos  
de  
programación

Carlos Andrés  
Delgado S.

Funciones  
recursivas  
auxiliares

Listas en listas

Abreviaciones  
de listas

Definiciones  
locales

## Definición

La función **insert** nos permite insertar un número en una lista ordenada de números

```
;;Propósito: Inserta un elementos en una lista ordenada  
;;insert: lista-de-numeros -> lista-de-numeros  
(insert 5 empty)  
->(cons 5 empty)  
(insert 10 (cons 2 (cons 11 (cons 14 empty))))  
-> (cons 2 (cons 10 (cons 11 (cons 14 empty))))
```

## Definición

### Ahora a definir la función **insert**

```
;;Propósito: Inserta un elementos en una lista ordenada
;;insert: numero, lista-de-numeros -> lista-de-numeros
(define (insert n lista)
  (cond
    [(empty? lista) (cons n empty)]
    [else ... (first lista) ... (insert n (rest lista)
    )
  )
)
```



## Definición

- 1 Se necesita verificar si  $n$  es mayor que ( $\text{car lista}$ ), si es así se inserta ( $\text{car lista}$ )
- 2 En caso contrario se inserta  $n$

# Funciones recursivas auxiliares

Fundamentos  
de  
programación

Carlos Andrés  
Delgado S.

Funciones  
recursivas  
auxiliares

Listas en listas

Abreviaciones  
de listas

Definiciones  
locales

## Definición

Como puede observar se necesita verificar

```
;;Propósito: Inserta un elementos en una lista ordenada
;;insert: numero, lista-de-numeros -> lista-de-numeros
(define (insert n lista)
  (cond
    [(empty? lista) (cons n empty)]
    [else
     (cond
      [(<= n (first lista)) (cons n lista)]
      [(> n (first lista))
       (cons (first lista)
              (insert n (rest lista)))]
      )
     ]
  )
)
```

# Funciones recursivas auxiliares

Fundamentos  
de  
programación

Carlos Andrés  
Delgado S.

Funciones  
recursivas  
auxiliares

Listas en listas

Abreviaciones  
de listas

Definiciones  
locales

"Carlos"

"Juan"

"999"

numero  $\rightarrow$  string  
Csymbol  $\rightarrow$  string  
~)

## Ejercicio

Un correo electrónico tiene la siguiente estructura:

```
(define-struct email (remitente destinatario mensaje))
```

Usted debe ordenar una lista de estas estructuras de acuerdo al **remitente**, utilice la función string<? para el ordenamiento

Fundamentos  
de  
programación

Carlos Andrés  
Delgado S.

Funciones  
recursivas  
auxiliares

Listas en listas

Abreviaciones  
de listas

Definiciones  
locales

1 Funciones recursivas auxiliares

2 Listas en listas

3 Abreviaciones de listas

4 Definiciones locales

## Definición

Las definiciones de listas que hemos visto hasta ahora son:

```
( cons n l )
```

Donde  $n$  es un valor (símbolo, booleano o número) y  $l$  es una lista con estructura **(cons n l)**.

# Listas en listas

Fundamentos  
de  
programación

Carlos Andrés  
Delgado S.

Funciones  
recursivas  
auxiliares

Listas en listas

Abreviaciones  
de listas

Definiciones  
locales

## Definición

Por ejemplo:

```
(cons (cons 1 (cons 2 empty)) empty)  
→(cons (cons 1 (cons 2 '())) '())
```

Esta es una lista que contiene una lista con los números 1 y 2.

# Listas en listas

Fundamentos  
de  
programación

Carlos Andrés  
Delgado S.

Funciones  
recursivas  
auxiliares

Listas en listas

Abreviaciones  
de listas

Definiciones  
locales

## Definición

Por ejemplo:

```
(cons (cons 'a (cons 'b empty)) empty)  
→(cons (cons 'a (cons 'b '())) '())
```

Esta es una lista que contiene una lista con los símbolos 'a y 'b.

## Definición

Para definir funciones para operar listas de listas, es necesario analizar si los elementos son una lista y tratarlos como listas.



## Definición

```
;;Función sumar lista
;;suma-lista: lista-de-listas-de-numeros -> numero
;;Esta función suma los elementos de una lista
(define
  (suma lista)
  (cond
    [(empty? lista) 0]
    [(cons? (first lista)) (+ (suma (first lista)) (suma
      (rest lista)))]
    [else (+ (first lista) (suma (rest lista)))]
  )
)
```

## Ejercicio

Diseñe una función que recibe una lista de lista de símbolos y retorna el número de elementos que contiene la lista, por ejemplo:

```
(size (cons (cons 'a (cons 'b empty)) empty))  
→ 2  
(size (cons (cons (cons 'x (cons 'y empty)) (cons 'b  
    empty)) (cons 'p empty)))  
→ 4
```

## Definición

Para crear pegar listas usted puede usar la función **append**:

```
(append (cons 1 (cons 2 empty)) (cons 3 empty))
```

Fundamentos  
de  
programación

Carlos Andrés  
Delgado S.

Funciones  
recursivas  
auxiliares

Listas en listas

Abreviaciones  
de listas

Definiciones  
locales

1 Funciones recursivas auxiliares

2 Listas en listas

3 Abreviaciones de listas

4 Definiciones locales

## Definición

Ahora vamos a cambiar de nivel :), toca seleccionar **nivel intermedio**

## Definición

Ahora vamos a utilizar una abreviación para la creación de listas con **list**

```
(cons 1 (cons 2 (cons 3 empty)))  
→(cons 1 (cons 2 (cons 3 '())))  
(list 1 2 3)  
→(cons 1 (cons 2 (cons 3 '())))  
(list 1 2 (list 1 2 3) 2)  
→(cons 1 (cons 2 (cons (cons 1 (cons 2 (cons 3 '()))) (cons 2 '()))))
```

## Ejercicio

Genere las abreviaciones con **list** para las siguientes listas:

- 1 `(cons (cons 'a (cons 'b empty)) empty)`
- 2 `(cons 2 (cons 5 (cons 7 empty)))`
- 3 `(cons (cons (cons 'x (cons 'y empty)) (cons 'b empty))  
(cons 'p empty))`
- 4 `(cons 4 (cons (cons 4 empty) (cons 4 empty)))`

Fundamentos  
de  
programación

Carlos Andrés  
Delgado S.

Funciones  
recursivas  
auxiliares

Listas en listas

Abreviaciones  
de listas

Definiciones  
locales

1 Funciones recursivas auxiliares

2 Listas en listas

3 Abreviaciones de listas

4 Definiciones locales



## Definición

Hasta ahora hemos realizado definiciones que son visibles en todo el programa, pero a veces queremos que estas sólo puedan ser vistas en algunas partes del código, para esto utilizamos la función **local**

## Definición

Por ejemplo:

```
(define a 4)
(local
  (
    (define b 6)
    (define c 7)
  )
  (+ a b c)
)
→ 17
```

En este caso `a` es visible en todo el programa, `b` y `c` son sólo visibles dentro de la estructura local.

## Definición

**local** presenta la siguiente estructura

```
(local  
  ( ... definiciones ... )  
  .. expresion ...  
)
```

## Definición

Esto nos permite calcular funciones y retornar argumentos automáticamente:

```
(define valor
  (local
    (
      (define (f x) (* x x))
      (define (g p w) (+ (f p) (f w)))
    )
    (g 4 5)
  )
)
(+ valor 4)
->45
```



Universidad  
del Valle

# ¿Preguntas?

Fundamentos  
de  
programación

Carlos Andrés  
Delgado S.

Funciones  
recursivas  
auxiliares

Listas en listas

Abreviaciones  
de listas

Definiciones  
locales

# VAMO A



# PROGRAMAR