

Fundamentos de programación

Procesamiento de datos simples III: Listas de tamaño fijo

Carlos Andrés Delgado S.

Facultad de Ingeniería. Universidad del Valle

Septiembre de 2018

Fundamentos
de
programación

Carlos Andrés
Delgado S.

Introducción a
las listas

Procesamiento
de listas

1 Introducción a las listas

2 Procesamiento de listas

Fundamentos
de
programación

Carlos Andrés
Delgado S.

Introducción a
las listas

Procesamiento
de listas

1 Introducción a las listas

2 Procesamiento de listas

Definición

En algunas ocasiones tenemos muchas piezas de información del mismo tipo y se tiene problemas en como procesarlas. Para esto se tiene un tipo de dato llamado **lista** que permite tener un número indefinido de piezas de información. Por ejemplo la lista de regalos de navidad de un niño:

```
"iphone 7" "plastilina" "sedante" "bandera pirata" "ojo  
de vidrio"
```

¿Cómo podemos manejar esta información?

Definición

Dr Racket ofrece una implementación para construcción de listas:

- 1 Para una lista vacía (sin elementos)

```
empty
```

make-list

- 2 Para una lista de un elemento:

```
(cons "iphone 7" empty)
```

- 3 Para una lista de dos elementos:

```
(cons "sedante" (cons "ojo de vidrio" empty))
```

¿Cómo construiría la lista de los regalos de navidad?:

```
"iphone 7" "plastilina" "sedante" "bandera pirata" "ojo de vidrio"
```

(cons "iphone 7"

(cons "plastilina"

(cons "sedante"

(cons "bandera pirata"

(cons "ojo de vidrio" empty)

)

)

)

)

Valor

lista

Definición

En general tenemos (**cons n v**)

- 1 n es un valor cualquiera: numero, booleano, símbolo, cadena de texto, estructura u otra lista
- 2 v es obligatoriamente una lista

Esto es una estructura, cuyo primer campo es un valor y el segundo campo es del mismo tipo de estructura.

Definición

La lista de nuestro niño es así:

```
(cons "iphone 7" (cons "plastilina" (cons "sedante" (cons  
"bandera pirata" (cons "ojo de vidrio" empty)))))
```

```
(cons 1 (cons 2 (cons 3 (cons 4 (cons 5 (cons 6  
  (cons 7 (cons 8 (cons 9 (cons 10 empty))))))))))
```

Ejercicios en clase

En Dr Racket:

- 1 Construya una lista con los planetas del sistema solar
- 2 Construya una lista con sus 4 comidas favoritas
- 3 Construya una lista con cuatro colores
- 4 Construya una lista con los números del 1 al 10

```
(cons 'mercurio (cons 'venus (cons 'tierra  
  (cons 'marte (cons 'jupiter (cons 'saturno  
    (cons 'urano (cons 'neptuno empty))))))))
```

Contrato

En nuestros contratos, ahora vamos a agregar el tipo de dato lista-de-X donde X son el tipo de dato de los elementos de la lista.

Ejemplo

```
;; Autor: Docente curso fundamentos de programación
;; Fecha de creación: 10-Septiembre-2016
;; Contrato: lista3numeros: numero,numero,numero -> lista-de-numeros
;; Propósito: Este programa recibe 3 números y retorna una lista
;; Ejemplo: (lista3numeros 1 2 3) retorna (cons 1 (cons 2 (cons 3 empty)))
;; Función
(define (lista3numeros x y z)
  (cons x (cons y (cons z empty))))
;;Ejemplo
(check-expect (lista3numeros 1 2 3) (cons 1 (cons 2 (cons 3 empty))))
```

Contrato

Los más comunes que vamos a usar

- 1 lista-de-numeros
- 2 lista-de-simbolos
- 3 lista-de-<estructura>, donde *estructura* es una estructura que usted ha construido

Ejercicios en clase

En Dr Racket:

- 1 Construya una función que reciba 3 colores y retorne una lista con ellos.
- 2 Construya una función que reciba 3 números (a, b y c):
 - 1 Si $a > c$ entonces retorna una lista en este orden ($a\ b\ c$)
 - 2 Si $a < c$ y $a < b$ entonces retorna una lista en este orden ($b\ c\ a$)
 - 3 En otro caso retorna ($c\ a\ b$)

Recuerde utilizar la receta de diseño

Acceder a una lista

Para acceder a una lista se cuentan con las siguiente funciones:

- 1 **first** Accede al primer elemento de la lista
- 2 **rest** Accede al resto de la lista
- 3 **length** Indica el tamaño de una lista

□ (first rest)
 ↑ ↑
 elemento lista

Acceder a una lista

Pruebe:

```
(define lista-numeros (cons 1 (cons 4 (cons 7 (cons 8  
    empty)))))  
(first lista-numeros)  
(rest lista-numeros)  
(first (rest lista-numeros))  
(rest (rest (lista-numeros)))
```

¿Que observa?

Acceder a una lista

Si observa detenidamente, `first` retorna elementos de la lista y `rest` listas, ya que la composición de una lista es un elemento seguido de una lista

```
(cons x y)  
;;x es elemento  
;;y es una lista que puede ser empty o otro cons
```


Acceder a una lista

Pruebe:

```
(define lista-numeros (cons 1 (cons 4 (cons 7 (cons 8  
    empty)))))  
(length lista-numeros)
```

¿Que observa?

Ejercicio

Dado el caso anterior:

`(define lista-numeros (cons 1 (cons 4 (cons 7 (cons 8
empty)))))`

(first rest rest lista-numeros)

- 1 Acceda al tercer elemento de la lista: debe retornar 7
- 2 Acceda al cuarto elemento de la lista: debe retornar 8
- 3 Acceda al último ~~elemento~~ de la lista: debe retornar empty o '()

Tenga cuidado, si ejecuta **(first empty)** o **(rest empty)** dará error ya que la lista vacía no tiene elementos.

(first (rest (rest (rest lista-numeros))))

(rest (rest (rest (rest lista-numeros))))

Fundamentos
de
programación

Carlos Andrés
Delgado S.

Introducción a
las listas

Procesamiento
de listas

1 Introducción a las listas

2 Procesamiento de listas

Ejercicios en clase

Para el procesamiento de listas debe tener en cuenta

- **first** permite acceder al primer elemento de la lista
- **rest** siempre retorna una lista, por lo que debe usarse **first** posteriormente para acceder al elemento
- La lista vacía **empty** no tiene elementos, por lo que puede preguntar con el predicado **empty?**
- Vamos a suponer que todos los elementos de la lista **son del mismo tipo**

Ejemplo

- 1 Construya una función que reciba una lista de 4 números y retorne su suma
- 2 Construya una función que recibe una lista de 4 símbolos y retorna verdadero si contiene el símbolo 'mula

```
;;Contrato: suma-lista: lista-numeros -> numero  
(define lista-numeros (cons 1 (cons 3 (cons 8 (cons  
  2 empty)))))  
  
;;Contrato: hay-mula?: lista-simbolos -> booleano  
(define lista-cosas (cons 'gato (cons 'perro (cons '  
  lobo (cons 'mula empty)))))
```

Recuerde utilizar la receta de diseño

Ejemplo

Construya una función que reciba una lista de 3 números y retorne su suma

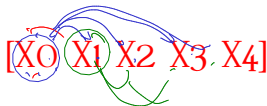
```
;;Autor: Docente curso fundamentos de programación
;;Fecha: 2-Septiembre-2018
;;Contrato: suma-lista: lista-numeros -> numero
;;Propósito: Suma los elementos de una lista de 4 números
;;Ejemplo (suma-lista (cons 1 (cons 3 (cons 8 (cons 2 empty))))) -> 14
(define (suma-lista l)
  (+
    (first l)
    (first (rest l))
    (first (rest (rest l)))
    (first (rest (rest (rest l)))))
  )
)
(define lista-numeros (cons 1 (cons 3 (cons 8 (cons 2 empty)))))
(check-expect (suma-lista lista-numeros) 14)
```

Ejemplo

Construya una función que recibe una lista de 4 símbolos y retorna verdadero si contiene el símbolo 'mula

```
;;Autor: Docente curso fundamentos de programación
;;Fecha: 2-Septiembre-2018
;;Contrato: hay-mula?: lista-simbolos -> booleano
;;Propósito: Indica si está el símbolo 'mula en una lista de 4 símbolos
;;Ejemplo (hay-mula? (cons 'gato (cons 'perro (cons 'lobo (cons 'mula empty))
))) -> True
(define (hay-mula? l)
  (cond
    [(equal? (first l) 'mula) true]
    [(equal? (first (rest l)) 'mula) true]
    [(equal? (first (rest (rest l))) 'mula) true]
    [(equal? (first (rest (rest (rest l)))) 'mula) true]
    [else false]
  )
)
(define lista-cosas (cons 'gato (cons 'perro (cons 'lobo (cons 'mula empty))
)))
(define lista-cosasB (cons 'gato (cons 'perro (cons 'lobo (cons 'paloma
empty)))))

(check-expect (hay-mula? lista-cosas) true)
(check-expect (hay-mula? lista-cosasB) false)
```



Ejercicios en clase

- 1 Crea una función **buscar-mayor** que recibe una lista de 5 números y retorna el mayor (todos son diferentes)
- 2 Crea una función **busca-simbolo** que recibe una lista de 5 símbolos y un símbolo, retorna verdadero si el símbolo esta en la lista y falso en caso contrario

Recuerde utilizar la receta de diseño

Ejercicio

Un producto tiene dos campos: un código (número) y un tipo (estructura tipo).

Un tipo tiene dos campos: nombre (símbolo) y categoría (símbolo).

Diseñe una función que reciba una lista de productos de tamaño 5, esta función retorna una lista que contiene las categorías de los tipos de cada elemento.



Universidad
del Valle

¿Preguntas?

Fundamentos
de
programación

Carlos Andrés
Delgado S.

Introducción a
las listas

Procesamiento
de listas

VAMO A



PROGRAMAR