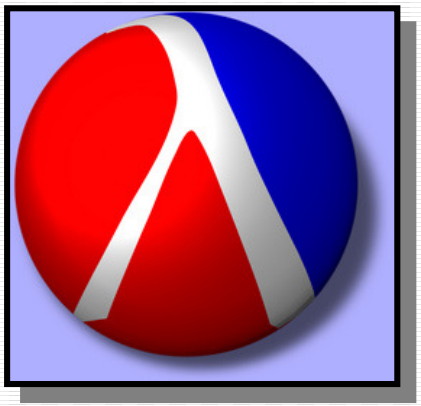
Fundamentos de Programación

Tipos de Datos: Listas



Profesor: Daniel Wilches Maradei
Diapositivas Originales: Jesús A. Aranda

Universidad del Valle

Introducción

- Las listas son “conjuntos” o agrupaciones de datos de cualquier tipo: estructuras, cadenas, símbolos, números, o incluso listas!

- Los elementos de una lista pueden ser de tipos diferentes, es decir, una lista puede contener como primer elemento una cadena y como segundo elemento un número

Introducción

- Las listas son usadas para almacenar datos cuando desconocemos, al momento de programar, la cantidad de datos que se guardarán.

- Las listas tienen longitud variable, y puede agregárseles elementos después de estar ya creadas.

Sintaxis y Ejemplo

- Podríamos crear una lista de estudiantes, con los nombres de los estudiantes de un curso:

```
(list "daniel" "alvaro" "pedro")
```

- Para crear una lista se usa la función predefinida list, seguida de todos los elementos que van a componerla.
- Otro ejemplo de lista podría ser:

```
(list "enero" 'febrero 15 2007)
```

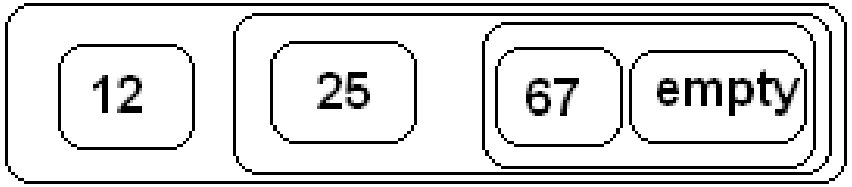
¿Qué es en verdad una lista?

- Para aprender a manejar bien las listas necesitamos conocer cómo están formadas: Una lista es verdad una sola pareja de elementos !!
- Entonces cómo pueden entonces guardarse varios elementos dentro de una lista? R:/ el segundo elemento de la pareja es otra lista.

Explicación didáctica

- Veamos la representación gráfica de la lista:

(list 12 25 67)



- Como pueden ver, la lista en verdad es una pareja, en la que el primer elemento es el 12, y el segundo elemento es la lista:

(list 25 67)

Explicación didáctica 2

(list 12 25 67) = (cons 12 (list 25 67))
(list 25 67) = (cons 25 (list 67))
(list 67) = (cons 67(list))
(list) = empty

Es decir:

(list 12 25 67) = (cons 12 (cons 25 (cons 67 empty)))

cons y empty

- cons es una función que sirve para crear parejas de datos.
- empty representa a una lista vacía (sin elementos)
- **Ahora si:** una lista es una pareja de datos en el cual el segundo dato es a su vez una lista que contiene todos los elementos de la lista original exceptuando el primero.

first y rest

- Para acceder a los elementos de una pareja usamos las funciones first y rest:
  - first me retorna el primer elemento de una lista (el primer elemento de la pareja)
  - rest, el resto de la lista (el segundo elemento de la pareja)

Un ejemplo completo usando listas

```
(define números (list 1 2 3 4 5 6))

(define (sumar-lista lon)
   (cond
        ((empty? lon) 0)
        (else (+ (first lon)
                 (sumar-lista (rest lon))))
   ))
```

Ejercicios en Clase

- Desarrollemos una función que reciba una lista de números y retorne el mayor número de esa lista.

- La función debe poder usarse así:

```
(mayor (list 1 4 2 5 7 4))
; retorna 7
```

Ejercicios en Clase

- Desarrollemos una función que reciba una lista de números y un número X, y retorne el primer elemento de la lista que sea mayor que X. Si no hay algún elemento mayor, debe retornar false.

- La función debe poder usarse así:

```
(mayor-que (list 1 4 2 5 7 4) 4)
; retorna 5
```

Ejercicios en Clase

- Desarrollemos una función que reciba una lista de números y un número N, y retorne el elemento N-ésimo de la lista. Si no hay existe tal elemento en la lista, debe retornar false.

- La función debe poder usarse así:

```
(list-get (list 1 4 2 5 7 4) 5)
; retorna 7
```

Ejercicios para la casa
(Así serán los del parcial)

1. Desarrolle una función que retorne el último número de una lista.
   La función debe poder usarse así:

```
(ultimo (list 1 4 3 2 5 7 4)) ; retorna 4
```

2. Desarrolle una función que retorne el promedio de los números de una lista.
   La función debe poder usarse así:

```
(promedio (list 3 2 5 7 8)) ; retorna 5
```

Ejercicios para la casa
(Así serán los del parcial)

3. Desarrolle una función que ordene los números de una lista.

   La función debe poder usarse así:

```
(ordenar (list 1 9 2 17 8))
; retorna (list 1 2 8 9 17)
```

4. Desarrolle una función que retorne el n-ésimo número más pequeño de una lista.

   La función debe poder usarse así:

```
(segundo-menor (list 11 3 2 7 4) 3)
; retorna 4
```

Ejercicios para la casa
(Así serán los del parcial)

5. Desarrolle una función que retorne true si un par de listas son iguales, false en caso contrario.
   La función debe poder usarse así:

```
(list=? (list 3 2 5 7 8) (list 3 2 5 7 3))
; retorna false
```

6. Desarrolle una función que retorne una lista con el resultado de sumar los elementos de 2 listas que recibe como parámetros.
   La función debe poder usarse así:

```
(list+ (list 5 2 15 7 8) (list 3 2 5 7 3))
; retorna (list 8 4 20 14 11)
```