



PARADIGMA FUNCIONAL - LISTAS

Profa. Maria Adelina Raupp Sganzerla
Paradigmas de Linguagem de Programação
Ulbra – Gravataí – 2016/2

LISTAS - SCHEME

- Lista é a estrutura de dados mais importante e poderosa que pode ser manipulada pela Linguagem;
- Representam sequência de valores;
- Exemplos:
 - A lista (10 20 40) tem 3 elementos
 - A lista ((10 20 40)) tem 1 elemento
 - A lista (10 20 40 (50 60)) tem 4 elementos



LISTAS - SCHEME

○ Algumas Primitivas para tratamento de Listas:

- **list**: cria uma lista a partir de argumentos de quaisquer tipos

```
> (list 1 2 3)  
(1 2 3)
```

- **car**: retorna o primeiro elemento da lista

```
> (car (list 1 2 3))  
1
```

- **cdr**: retorna a lista sem o primeiro elemento

```
> (cdr (list 1 2 3))  
(2 3)
```



LISTAS - SCHEME

○ Composto Primitivas:

- Como fazer para extrair o segundo elemento da lista (1 2 3)?

```
> (car (cdr (list 1 2 3)))  
(2)
```

- ... e para retornar a lista (1 2 3) somente com o terceiro elemento?

```
> (cdr (cdr (list 1 2 3)))  
(3)
```



LISTAS - SCHEME

- Composto Primitivas:

- Como criar a lista ((1 2) 3)?

```
> (list (list 1 2) 3)  
((1 2) 3)
```

- Como esvaziar uma lista?

```
> (cdr (list 3))  
()
```



LISTAS - SCHEME

- Algumas Primitivas para tratamento de listas

- **cons**: acrescenta o elemento a no início da lista

```
> (cons 1 (list 2 4))  
(1 2 4)
```

- **append**: toma duas listas como argumentos e forma uma lista única

```
> (append (list 2 3) (list 5 6))  
(2 3 5 6)
```



PREDICADOS - SCHEME

- Predicados são funções que retornam Verdadeiro ou Falso

<code>(< 5 6)</code>	<code>(<= 8 6)</code>	<code>(= 4 4)</code>
True (#t)	False	True (#t)

- **null?**: retorna True se o argumento é nulo, senão retorna False (nil); serve, também, como negação lógica

```
(null? (cdr (list 1)))  
True
```

- **eq?**: retorna True, se os argumentos simbólicos são iguais, senão, retorna nil

<code>(eq? 'a 'b)</code>	<code>(eq? 'a 'a)</code>
False	True (#t)



PREDICADOS - SCHEME

○ Outros Predicados:

- **number?:** Testa se um dado valor é um número;
- **symbol?:** Testa se um dado valor é um símbolo;
- **list?:** Testa se um dado valor é uma lista ;
- **even?:** Testa se um dado número inteiro é par;
- **odd?:** Testa se um dado número inteiro é ímpar;
- **positive?:** Testa se um dado número é maior que 0;
- **zero?:** Testa se um dado número é igual a 0.



CONDICIONAIS - SCHEME

- Comando **if**:

```
(define (maximo x y) (if (> x y) x y))  
  
      (maximo 5 6)                (maximo 6 5)  
              6                      6
```

- Comando **cond**: generalização do if para vários casos

```
(define (sinal x) (cond ((< x 0) -1)  
                        ((> x 0) 1)  
                        (else 0)))
```

```
(sinal 4)    (sinal -3)    (sinal 0)  
    1          -1          0
```



OPERADORES LÓGICOS - SCHEME

- Comando **and**:

```
(and (= 1 1) (> 5 4))  
true
```

- Comando **or**:

```
(or (= 1 2) (> 5 4))  
true
```

- Comando **not**:

```
(not (< 5 0))  
true
```



RECURSÃO

- Cálculo do Fatorial

$$n! = \begin{cases} 1, & \text{se } n = 0 \\ n * (n-1)!, & \text{caso contrário} \end{cases}$$

```
(define (fatorial n)
  (if (zero? n)
      1
      (* n (fatorial (- n 1)))))
```

```
(fatorial 4)
```

24



LAÇOS DE REPETIÇÃO - RECURSIVIDADE

- Exemplo: Soma dos elementos de uma lista

```
(define (soma_aux lista sm)
  (if (null? lista)
      sm
      (soma_aux (cdr lista) (+ sm (car lista))))))
```

```
(define (soma lista) (soma_aux lista 0))
```

```
(soma (list 2 3 5))
10
```

