



EA 072 Inteligência Artificial em Aplicações Industriais

LISP

Lisp

- Programas

- <http://www.dca.fee.unicamp.br/courses/EA072/software/>

- Interpretador (Avaliador)

- Read / eval / print

- notação pré-fixa

- (<operador> <par.1> ... <par.n>)

■ Quote

– Evita que o símbolo seja avaliado

– Abreviação: 

- `> (quote a)`
- `A`
- `> 'a`
- `A`
- `> A`
- `error: unbound variable - A`
- `> (setq a (1 2))`
- `error: bad function - 1`

■ Símbolos

– Exemplos:

- `c1`
- `bar`
- `baaz-quux-garply`

– Keywords

- `:this-is-a-keyword`

■ Números

– Exemplos:

- `-34`
- `+6`
- `3.1415`
- `1.722e-15`
- `#c(1.722e-15 0.75)`

■ Estruturas do tipo `conses`

– `car` & `cdr`

- `> (cons 4 5)`
- `(4 . 5)`
- `> (cons (cons 4 5) 6)`
- `(4 . 5 . 6)`
- `> (car (cons 4 5))`
- `4`
- `> (cdr (cons 4 5))`
- `5`

■ Listas

– “Conse” terminada em nil

- `> (cons 4 nil)`
- `(4)`
- `> (cons 4 (cons 5 (cons 6 nil)))`
- `(4 5 6)`
- `> (list 4 5 6)`
- `(4 5 6)`
- `> (car (4 5 6))`
- `4`

■ Funções

– Exemplos:

- `> (+ 3 4 5 6)`
- `18`
- ```
>(defun fact (x)
 (if (> x 0)
 (* x (fact (- x 1)))
 1)
))
```

```
FACT
> (fact 5)
120
```

### – Parâmetros opcionais (&optional)

- `(defun bar(x &optional y)(if y x 0))`

## ■ Variáveis

- `> (setq x 7) ;(setf x 7)`
- `> x`
- `7`
- `(let ((var1 val1) (var2 val2))  
    (body)  
)`
- `> (let ((x 1) (y (+ x 1))) y)`
- `8`
- `(let* ((x 1) (y (+ x 1))) y)`
- `2`



## ■ Array

- ```
>(setq a (make-array '(4 3)  
:initial-contents  
'( (1 2 3)  
    (4 5 6)  
    (7 8 x)  
    (2 2 0) )))  
#2a((1 2 3)(4 5 6)(7 8 x)(2 2 0))
```
- ```
> (aref a 2 2)
```
- ```
x
```
- ```
(setf (aref a 2 2) y)
#2a((1 2 3)(4 5 6)(7 8 y)(2 2 0))
```

## ■ Condicionais

- (if <condição> <then> <else>)
  - > (if t 2 0)
  - 2
  - > (if nil 2 0)
  - 0
  - (if (< a b) (+a b) (- a b))
- (when <condição> <ação>)
- (unless <condição> <ação>)
- (cond (condição-1 expressão-11...expressão-1l)  
(condição-2 expressão-21...expressão-2m)  
...  
(condição-n expressão-n1...expressão-nk))

## ■ Sequenciação

```
- prog1 & progn
- (if (> 3 2)
 (prog1
 (+ 2 3)
 (print 'estou-no-consequente))
 (prog1
 (* 4 5)
 (print 'estou-na-alternativa)))
```