Paradigmas de Linguagens de Programação - 2012.1 - Prof. Sergio D. Zorzo
Aula 04 - Lisp - Exemplos

Exemplo 1
Uso do lispworks

1. Abrir o LispWorks, e executar os seguintes comandos no listener

```
> (+ 2 3)
> (/ 10 2)
```

2. Criar um novo arquivo

```
(defun quadrado (x) (* x x))
```

3. Compilar e executar no listener

```
> (quadrado 10)
```

Exemplo 2
Avaliação de expressões

1. Abrir o LispWorks, e executar os seguintes comandos no listener

```
>(* (+ 2 5) (- 7 (/ 21 7)))
>(= (+ 2 3) 5)
>(* 2 x)
```

Exemplo 3
Comandos de lista e quote

1. Abrir o LispWorks, e executar os seguintes comandos no listener

```
> (list 1 2 3 4 5)
> (list (+ 1 2) (+ 3 4))
```

2. Demonstrar o uso do quote

```
> (quote (a b c))
> (quote (+ 1 3))
> '(a b c)
> '(+ 1 3)
> '(- (+ 3 4) 7)
> (list (+ 1 2) (+ 3 4))
> (list '(+ 1 2) '(+ 3 4))
```

3. Demonstrar outros comandos de lista

```
> (nth 2 '(a b c d))
> (nth 1 (list 1 2 3 4 5))
> (nth 2 '((a 1) (b 2) (c 3) (d 4)))
> (length '(a b c d))
> (length '(1 2 (3 4) 5 6))
> (member a '(a b c d e))
```

```
> (member 'a '(a b c d e))
> (member 'c '(a b c d e))
> (member 5 '(1 2 3 4 5))
> (member 'a '(1 2 3 4 5))
```

4. Demonstrar CAR e CDR

```
> (car '(a b c))
> (cdr '(a b c))
> (car '((a b) (c d)))
> (car (cdr '(a b c d)))
> (car (car (cdr (cdr '(HOJE E (DIA DE) AULA)))))
> (caaddr '(HOJE E (DIA DE) AULA))
```

5. Demonstrar CONS

```
> (CONS '(A B C) '(A B C))
> (CONS 'NADA ( ))
> (CONS '((PRIM SEG) TER) '( ))
```

6. Demonstrar APPEND

```
> (APPEND '(A B) '(C))
> (APPEND '(A) '( ) '(B) '( ))
> (LIST '(A) '( ) '(B) '( ))
```

Exemplo 4
Comandos de atribuição

1. Abrir o LispWorks, e executar os seguintes comandos no listener

```
> (setq L '(A B))
> (setq L1 '(A B) L2 '(C D) N 3)
> L
> (car L2)
```

2. Demonstrar efeito colateral

```
> (defun contador () (SETQ X (+ X 1)))
> (SETQ X 1)
> (contador)
> (contador)
> (contador)
```

Exemplo 5
Definição de novas funções

1. Abrir o LispWorks, e executar os seguintes comandos no listener

```
> (defun quadrado (x) (* x x))
```

2. Demonstrar a chamada

```
> (quadrado 5)
> (quadrado (+ 5 5))
> (quadrado (quadrado 2))
```

Exemplo 6
Alguns predicados do LISP

1. Abrir o LispWorks, e executar os seguintes comandos no listener

```
> (atom 5)
> (atom 'L)
> (setq L '(a b c))
> (atom L)
> (listp '(a b c))
> (setq L '(a b c))
> (listp L)
> (listp 'L)
> (equal 'a 'a)
> (equal L L)
> (equal L '(a b))
> (equal L 'L)
> (null '( ))
> (null L)
> (null 'L)
> (numberp 5)
> (setq N 5)
> (numberp N)
> (setq Digitos '(1 2 3 4 5))
> (numberp Digitos)
> (setq zero 0)
> (zerop zero)
> (setq N -5)
> (minusp N)
```

Exemplo 7
Condicional

1. Abrir o LispWorks, e executar os seguintes comandos no listener

```
> (defun valor-absoluto (x)
        (cond ((< x 0) (- x))
              ((>= x 0) x)))
> (valor-absoluto 5)
> (valor-absoluto -5)
> (defun valor-absoluto (x)
        (cond ((< x 0) (- x))
              (t x)))
```

2. Demonstrar o If

```
> (defun nummax ( a b c)
     (if (> a b)
          (if (> a c)  a  c )
          (if (> b c)  b  c ) )  )
> (nummax 10 15 12)
```

Exemplo 8
Igualdade

1. Abrir o LispWorks, e executar os seguintes comandos no listener

```
> (eq 'a 'b)
> (eql 'a 'b)
> (equal 'a 'b)
> (equalp 'a 'b)

> (eq 'a 'a)
> (eql 'a 'a)
> (equal 'a 'a)
> (equalp 'a 'a)

> (eq 3 3)
> (eql 3 3)
> (equal 3 3)
> (equalp 3 3)

> (eq 3 3.0)
> (eql 3 3.0)
> (equal 3 3.0)
> (equalp 3 3.0)

> (eq 3.0 3.0)
> (eql 3.0 3.0)
> (equal 3.0 3.0)
> (equalp 3.0 3.0)

> (eq '(a b c) '(a b c))
> (eql '(a b c) '(a b c))
> (equal '(a b c) '(a b c))
> (equalp '(a b c) '(a b c))

> (eq (cons 'a 'b) (cons 'a 'c))
> (eql (cons 'a 'b) (cons 'a 'c))
> (equal (cons 'a 'b) (cons 'a 'c))
> (equalp (cons 'a 'b) (cons 'a 'c))

> (eq (cons 'a 'b) (cons 'a 'b))
> (eql (cons 'a 'b) (cons 'a 'b))
> (equal (cons 'a 'b) (cons 'a 'b))
> (equalp (cons 'a 'b) (cons 'a 'b))

> (eq "Foo" "Foo")
> (eql "Foo" "Foo")
> (equal "Foo" "Foo")
> (equalp "Foo" "Foo")

> (eq "Foo" (copy-seq "Foo"))
> (eql "Foo" (copy-seq "Foo"))
> (equal "Foo" (copy-seq "Foo"))
> (equalp "Foo" (copy-seq "Foo"))

> (eq "FOO" "foo")
> (eql "FOO" "foo")
> (equal "FOO" "foo")
> (equalp "FOO" "foo")
```

Exemplo 9
Conectivos lógicos

1. Abrir o LispWorks, e executar os seguintes comandos no listener

```
> (and t t)
> (or t nil)
> (and t t t t t nil t)
> (or nil nil nil t nil nil)
> (and 2 4)
> (and '(a b) '(c d))
> (or '() nil)
> (or '() t 2)
> (or '() 2 t)
> (defun verdadeiro () (print 'aqui) t)
> (or '() (verdadeiro) t)
> (or '() t (verdadeiro))
> (and t 2 (verdadeiro) nil)
> (and t 2 nil (verdadeiro))
```

Exemplo 10
Variáveis livres e ligadas

1. Abrir o LispWorks, e executar os seguintes comandos no listener

```
> (defun incremento (parametro)
  (setq parametro (+ parametro livre))
   (setq saida parametro))
> (incremento 10)
> (setq livre 5)
> (incremento 10)
> (setq parametro 1000)
> (incremento 10)
```

Exemplo 11
Let e let*

1. Abrir o LispWorks, e executar os seguintes comandos no listener

```
> (let ((x 3)
        (y 4))
    (print (+ x y)))
> x
> y
> (setq x 20)
> (let ((x 3)
        (y 4))
    (print (+ x y)))
> x
```

2. Demonstrar o uso de let*

```
> (makunbound 'x) ; caso precise remover a variável da memória global
> (let ((x 2)
        (y (+ x 1)))
     (print y))
> (let* ((x 2)
        (y (+ x 1)))
     (print y))
```

3. Demonstrar o uso de psetq

```
> (makunbound 'x)
> (makunbound 'y)
> (setq x 1 y (+ x 1))
> x
> y
> (makunbound 'x)
> (makunbound 'y)
> (psetq x 1 y (+ x 1))
```

3.1. Como trocar os valores de duas variáveis facilmente

```
> (setq a 1 b 2)
> a
> b
> (psetq a b b a)
> a
> b
> (setq a b b a)
> a
> b
```

Exemplo 12
Passagem por valor

1. Abrir o LispWorks, e executar os seguintes comandos no listener

```
> (defun proc (param)
      (setq param (+ param 1)))
> (proc 2)
> (proc 40)
> (setq x 10)
> (proc x)
> x
```

Exemplo 13
Passagem de funções como parâmetros

1. Abrir o LispWorks, e criar um novo programa

```
(defun soma (x y)
  (print 'soma)
  (+ x y))
(defun sub (x y)
  (print 'sub)
  (- x y))
(defun mult (x y)
  (print 'mult)
  (* x y))
(defun executa (funcao args)
  (apply funcao args))
(defun executa2 (funcao arg1 arg2)
  (funcall funcao arg1 arg2))
```

2. Testar os comandos no listener

```
> (executa 'soma '(10 5))
> (executa2 'mult 3 7)
```

Exemplo 14
When e unless

1. Abrir o LispWorks, e criar um novo programa

```
(defun calculo (x y z)
  (if (not (= y 0))
      (progn (setq x (/ 100 y)) (setq z (+ x 20)) (* z 100))
   nil))
```

2. Testar os comandos no listener

```
> (calculo 10 20 30)
> (calculo 10 0 30)
```

3. Mostrar com When

```
(defun calculo (x y z)
  (when (not (= y 0)) (setq x (/ 100 y)) (setq z (+ x 20)) (* z 100)))
```

4. Mostrar com Unless

```
(defun calculo (x y z)
  (unless (= y 0) (setq x (/ 100 y)) (setq z (+ x 20)) (* z 100)))
```

Exemplo 15
Do, Dolist, Dotimes e Mapcar

1. Abrir o LispWorks, e criar um novo programa

```
(defun tira_elem (lista elem)
  (do ((l-aux lista (cdr l-aux))
       (res '()))
      ((null l-aux) res)
    (if (not (equal (car l-aux) elem))
        (setq res (cons (car l-aux) res)))))
```

2. Testar os comandos no listener

```
> (tira_elem '(a b c a c b) 'b)
```

3. Demonstrar dolist. Primeiro executar no listener os seguintes comandos

```
> (dolist (x '(1 2 3)) (print x))
> (dolist (x '(1 2 3 4 5)) (if (oddp x) (print x)))
```

4. Refazer o programa anterior, para usar dolist

```
(defun tira_elem (lista elem)
  (let ((res '()))
  (dolist (e lista res)
    (if (not (equal e elem))
        (setq res (cons e res)))))
```

```
)
```

5. Testar no listener

```
> (tira_elem '(a b c a c b) 'b)
```

6. Demonstrar dotimes

```
> (dotimes (i 4) (print i) )
> (dotimes (i 4 'fim) (print i))
```

7. Demonstrar mapcar

```
> (mapcar 'abs '(1 2 -10 -20 5 -2))
> (mapcar '+ '(1 2 3 4 5) '(5 4 3 2 1))
> (mapcar 'list '(a b c d) '(1 2 3 4))
```

Exemplo 16
Entrada e saída

---

1. Abrir o LispWorks, e criar um novo programa

```
(defun programa ()
  (let ((nome nil)
        (sobrenome nil))
    (print "Bem-vindo ao programa")
    (terpri)
    (print "Digite seu nome:")
    (setq nome (string (read)))
    (print "Digite seu sobrenome:")
    (setq sobrenome (string (read)))
    (print "Formato científico:")
    (princ (formatoCientifico nome sobrenome))
    (fresh-line))
  nil
)

(defun formatoCientifico (nome sobrenome)
  (concatenate 'string (string-upcase sobrenome) ", " (string (char nome
0)))
)
```

2. Testar no listener

```
> (programa)
```

Exemplo 9
Escopo léxico vs escopo dinâmico

---

1. Abrir o LispWorks, e rodar no listener
1.1. Exemplo de variável léxica definida com let

```
> (let ((x 1))
    (print x)
      (let ((x 2))
        (print x)
```

```
        (setq x 3)
        (print x))
    (print x))
```

1.2. Exemplo de variável léxica definida com setq

```
> (setq var-lex 5)
> (defun verifica-lex() var-lex)
> (verifica-lex)
> (let ((var-lex 6)) (verifica-lex))
```

1.3. Exemplo de variável dinâmica definida com defvar

```
> (setq x 1)
> (defun baz () x)
> (defun foo ()
    (let ((x 2))
      (baz)))
> x
> (baz)
> (foo)
> (defvar x 1)
> x
> (baz)
> (foo)
```

1.4. Exemplo de variável dinâmica definida com declare special
1.5. Reiniciar o lispworks para “apagar” as variáveis

```
> (defun baz () x)
> (defun foo ()
    (let ((x 2))
      (baz)))
> (foo)
> (defun foo ()
    (let ((x 2))
      (declare (special x))
      (baz)))
> (foo)
```