Programação Funcional – INE5363

Métodos e Técnicas de Programação Funcional: Linguagem LISP

Resolução de Exercício de LISP 2

http://www.inf.ufsc.br/~awangenh/Funcional/lisp2.html

Por: Léo Willian Kölln

Email: leokolln@gmail.com

Ciências da Computação – CCO
Departamento de Informática e Estatística – INE
Centro Tecnológico – CTC
Universidade Federal de Santa Catarina – UFSC

Florianópolis – Santa Catarina – Brasil
26 de Junho de 2006

3.2.18.1. Escreva uma declaração em LISP para executar cada uma das operações abaixo:

Nestes exercícios estarei evitando ao máximo usar "print", assim, quando for pedido para imprimir ou exibir na tela, estarei considerando o resultado produzido na tela ao invocar tal função. Porém deve se estar ciente de que caso levado o termo imprimir ao pé da letra como tendo que usar a função "print", uma função como "(defun glista (x) (list x))" deveria ser "(defun glista (x) (progn (print (list x)) nil)".

- **Ler dois números, imprimir sua soma e acrescentar 3 ao resultado. Assim 5 e 11 devem produzir 16 e 19 na tela.**

```
(defun soma3 (x y) (progn (print (+ x y)) (+ x y 3)))
```

- **Ler um único valor e imprimí-lo como uma lista. Assim o valor 6 deve produzir (6).**

```
(defun glista (x) (list x))
```

- **Ler dois valores e imprimir sua soma como uma lista. Deste modo 6 e 7 devem produzir a lista (13).**

```
(defun somalista (a b) (list(+ a b)))
```

Poderia também ser usada a função definida anteriormente, assim ficaria:

```
(defun somalista (a b) (glista(+ a b)))
```

- **Ler três números e imprimí-los como uma lista.**

```
(defun glista3 (a b c) (list a b c))
```

- **Ler três números e imprimir a soma dos dois primeiros e o produto desta pelo terceiro como uma lista.**

```
(defun multilista (x y z) (print (+ x y)) (list (* (+ x y) z)))
```

Acredito que o enunciado possa ter uma segunda interpretação, sendo que neste caso a seguinte definção não estaria errada:

```
(defun multilista (x y z) (list (* (+ x y) z)))
```

3.2.18.2. Escreva uma função que:

- **Devolva o valor 1 se seu parâmetro for maior que zero, -1 se for negativo, 0 se for zero.**

```
(defun sinal (x) (if (= x 0) x (if (< x 0) -1 1)))
```

- **Leia um nome. Se este for o mesmo nome que o dado como parâmetro, a função deve imprimir uma saudação simplese devolver o valor t. Se for diferente, nao deve imprimir nada e devolver nil.**

```
(defun testanome (n) (if (equal n "Bob") (progn (print "Nome
correto, é Bob") n) nil))
```

- **Dados três parâmetros, se o primeiro for um asterisco, os outros dois serão multiplicados; se for uma barra, o segundo deve ser dividido pelo terceiro; se não for nenhum dos dois, imprima uma mensagem de erro e assuma o valor zero. A função deve devolver como valor o resultado da operação aritmética.**

```
(defun operador (o a b) (if (equal o *) (* a b) (if (equal o /) (/ a b) (progn (print "A operação não está definida.") 0))))
```

- **Devolva t se seu primeiro parâmetro estiver no conjunto de valores especificado pelo seu segundo e terceiro parâmetros e nil se não estiver. Asim: (func-4 5 5 7) = t e (func-4 6 5 7) = nil.**

```
(defun verifica (x a b) (if (= x a) t (if (= x b) t nil)))
```

- **Aceite um valor simples e uma lista como parâmetros. Devolva t se o valor estiver na lista, nil caso nao esteja (este exercício pode ser resolvido de forma recursiva – pense um pouco...).**

```
(defun verificalista (x l) (let ((lp (pop l))) (if (equal lp nil) nil (if (= x lp) t (verificalista x l)))))
```