Paradigmas de Programação – Exercícios

28 de maio de 2010

Ex. 1 — Converta as seguintes expressões para a notação prefixa, e verifique o resultado (você pode digitar as expressões *infixas* tanto em Matlab/Octave, ou algum outro programa do gênero e comparar com o o resultado do interpretador Scheme para a versão *prefixa*).

a) $a + b + c + d + e + f$

b) $a + b - \frac{1}{(c-b)}$

c) $a + 1 - b - 2 + c + 3$

d) $\frac{(a+b)(c+d)}{e+1}$

e) $\frac{(a+b+c)(d+e+f)}{g+1}$

f) $\frac{2a}{b-c}$

Ex. 2 — Converta as seguintes expressões para a notação infixa, e verifique se sua conversão foi correta.

a) `(+ (* a b c) d e f)`

b) `(+ (- a b) c d)`

c) `(* a b (+ c d) (- e f) g)`

d) `(* 2 a (/ b 4) (+ c 10))`

e) `(/ 1 (+ x y z))`

f) `(* (+ a b c) (* d e (+ f g) (- h i)))`

Ex. 3 — Explique cuidadosamente o que significam as expressões e quais seus valores:

a) `(lambda (x y) (x y))`

b) `((lambda (x y) (x y)) display display)`

c) `((lambda (x y) (x y)) display 'display)`

Ex. 4 — Faça uma função Scheme que receba:

- Uma outra função (não interessa o número de argumentos)
- Uma lista de argumentos
- Um valor

e em seguida aplique a função com os argumentos, verifique se os argumentos são iguais ao valor (retornando `#t`) ou não (rtornando `#f`). Por exemplo, se eu fiz uma função soma,
`(testa soma 9 '(2 3 4))`
deve retornar `#t`.

Ex. 5 — Mude a função do exercício anterior. Quando o valor retornado for um número, a função de teste deverá tolerar um desvio de 0.01% no valor retornado.

```
(testa soma 9 '(2.0001 3 4.001))

  ==> #t
```

Ex. 6 — Faça uma função que receba uma lista de funções, valores e argumentos e teste uma por uma, mostrando quais não passam no teste. (Isso já é o *core* de um sistema de testes unitários!)