Trabalho 2 - PLP : Programas usando Common Lisp

João Vitor Azevedo Marciano 743554
Vitor Hugo Chaves Cambui 744358

June 2019

Conteúdo

1 Objetivo **2**

2 Codigo-Fonte **2**
2.1 Desparentize 2
2.2 Conta Atomos 3
2.3 monta-pares (Exercicio 1) 3
2.4 Main 3
2.5 monta_pares (Exercicio 2) 3
2.6 coleciona_repeticoes 3
2.7 elimina_repeticoes 3

3 Casos de Teste **5**
3.1 Caso Exercicio 1 5
3.2 Caso Exercicio 2 6

1 Objetivo

Escreva programas Lisp para resolver os seguintes problemas:
1) Dada uma lista L que contém elementos de qualquer tipo, possivelmente com repetições, construir outra lista que mostre quantas vezes cada elemento atômico (átomo, número, lista vazia) aparece na lista dada, inclusive nas sublistas. A lista resultante deve conter pares de elementos (pares são listas de dois elementos) sendo o primeiro elemento do par um elemento atômico que aparece na lista dada e o segundo elemento do par, o número de vezes que esse elemento aparece na lista. A ordem dos elementos na lista original deve ser mantida.

Por exemplo, dada a lista
L = (a b z x 4.6 (a x) () (5 z x) ())
Deve ser construída a lista
((a 2) (b 1) (z 2) (x 3) (4.6 1) (() 2) (5 1))
(Sugestão: desparentizar a lista dada antes de fazer as outras operações.)

2) Dada uma lista L com elementos de qualquer tipo, construir outra lista em que repetições consecutivas de elementos devem ser substituídas por pares da forma (N E), onde N é o número de repetições consecutivas do elemento E.

Por exemplo, dada a lista
L = (a a a a b c c a a d e e e e)
Deve ser construída a lista
((4 a) (1 b) (2 c) (2 a) (1 d) (4 e))

2 Codigo-Fonte

Para código mais "copiável" visitar esse link

2.1 Desparentize

```
;funcao que deixa todos os atomos no mesmo nivel
(defun desparentize(lista)
(cond ((null lista) nil)
((atom (car lista)) (cons (car Lista) (desparentize (cdr lista))))
((listp (car lista)) (append (desparentize (car lista)) (desparentize (cdr lista))))))
```

2.2 Conta Atomos

;recebe um elemento e uma lista, e retorna quantas vezes esse elemento aparece nela (defun conta-atomos(Elem Lista)
(cond ((null Lista) 0)
((equal Elem (car Lista)) (+ 1 (conta-atomos Elem (cdr Lista))))
(t (conta-atomos Elem (cdr Lista)))))

2.3 monta-pares (Exercicio 1)

;Monta os pares conforme o enunciado = (Elem nroRepeticoes)
(defun monta-pares(Elem Lista)
(list Elem (conta-atomos Elem Lista)))

2.4 Main

;Função principal (defun main(Lista)
(setq Lista1 (desparentize Lista))
(cond ((null Lista1) nil)
(t (cons (monta-pares (car Lista1) Lista1) (main (remove (car Lista1) (cdr Lista1)))))))

2.5 monta_pares (Exercicio 2)

;Função que serve para construir a lista de pares da forma (número_de_repetições elemento) de forma recursiva.
(defun monta_pares (lista)
(if (equal lista NIL)
NIL
(cons (list (length (coleciona_repeticoes lista)) (car lista)) (monta_pares (elimina_repeticao lista)))))

2.6 coleciona_repeticoes

;Função que serve para construir uma lista com todas as repetições consecutivas de um mesmo elemento (defun coleciona_repeticoes(lista)
(cond ((equal lista NIL) NIL)
((equal (cdr lista) NIL) lista)
((equal (car lista) (cadr lista))
(cons (car lista) (coleciona_repeticoes (cdr lista))))
(t (list (car lista)))))) newpage

2.7 elimina_repeticoes

;Função que serve para remover da cabeça da lista uma sequência de elementos que se repetem consecutivamente

```
(defun elimina_repeticao(lista)
(cond ((equal lista NIL) NIL)
((equal (cdr lista) NIL) NIL)
((equal (car lista) (cadr lista))
(elimina_repeticao (cdr lista)))
(t (cdr lista))))
```

3 Casos de Teste

3.1 Caso Exercicio 1

Para L = (a b z x 4.6 (a x) () (5 z x) ())
Deve ser construída a lista ((a 2) (b 1) (z 2) (x 3) (4.6 1) (() 2) (5 1))

Language: Common Lisp ▾ Editor: CodeMirror ▾ Layout: Vertical ▾

```
21      )
22  )
23  ;recebe um elemento e uma lista, e retorna quantas vezes esse elemento aparece nela
24  (defun conta-atomos(Elem Lista)
25      (cond ((null Lista) 0)
26            ((equal Elem (car Lista)) (+ 1 (conta-atomos Elem (cdr Lista))))
27            (t (conta-atomos Elem (cdr Lista)))
28
29      )
30  )
31  ;Monta os pares conforme o enunciado = (Elem nroRepeticoes)
32  (defun monta-pares(Elem Lista)
33      (list Elem (conta-atomos Elem  Lista))
34  )
35
36  ;Funcao principal
37  (defun main(Lista)
38      (setq Lista1 (desparentize Lista))
39      (cond ((null Lista1) nil)
40            (t (cons (monta-pares (car Lista1)  Lista1)
41                                  (main (remove (car Lista1) (cdr Lista1)))))
42      )
43  )
44
45
46  (print "Execução exerc1 para a lista dada (a b z x 4.6 (a x) () (5 z x) ())")
47  (print (main  '(a b z x 4.6 (a x) () (5 z x) ()))  )
48
49
50
```

Run it (F8) Save it [+] Show input Live cooperatio

Absolute running time: 0.08 sec, cpu time: 0.02 sec, memory peak: 5 Mb, absolute service time: 0,08 sec

```
"Execução exerc1 para a lista dada (a b z x 4.6 (a x) () (5 z x) ())"
((A 2) (B 1) (Z 2) (X 3) (4.6 1) (NIL 2) (5 1))
```

3.2 Caso Exercicio 2

Para L = (a a a a b c c a a d e e e e)
Deve ser construída a lista
((4 a) (1 b) (2 c) (2 a) (1 d) (4 e))

Language: Common Lisp ▾ Editor: CodeMirror ▾ Layout: Vertical ▾

```
            NIL
            (cons (list (length (coleciona_repeticoes lista)) (car lista)) (monta_pares (elimina_repeticao lista)))
          )
)

;Função que serve para construir uma lista com todas as repetições consecutivas de um mesmo elemento
(defun coleciona_repeticoes(lista)
          (cond ((equal lista NIL) NIL)
              ((equal (cdr lista) NIL) lista)
              ((equal (car lista) (cadr lista))
                (cons (car lista) (coleciona_repeticoes (cdr lista))))
               (t (list (car lista)))
          )
)

;Função que serve para remover da cabeça da lista uma sequência de elementos que se repetem consecutivamente
(defun elimina_repeticao(lista)
          (cond ((equal lista NIL) NIL)
              ((equal (cdr lista) NIL) NIL)
              ((equal (car lista) (cadr lista))
                (elimina_repeticao (cdr lista)))
               (t (cdr lista))
          )
)

;monta_pares do Exercicio 2: não confundir com 'monta-pares'  "
(print "Execução exerc2 para a lista dada (a a a a b c c a a d e e e e)")
(print (monta_pares  '(a a a a b c c a a d e e e e) ))
```

Run it (F8) Save it [+] Show input Live cooperation

Absolute running time: 0.08 sec, cpu time: 0.02 sec, memory peak: 5 Mb, absolute service time: 0,1 sec

```
"                                                                    "
"Execução exerc2 para a lista dada (a a a a b c c a a d e e e e)"
((4 A) (1 B) (2 C) (2 A) (1 D) (4 E))
```

Referências

Robert W. Sebesta *Conceitos de Linguagens de Programacao.*