Paradigma de Programação Funcional

6.0 - Funções - Exercícios

6.0.1) Analise os exemplos e exercícios de dados compostos e redefina as funções (que for possível) utilizando as funções `foldr`, `map` e `filter` (e outras funções de alta ordem). As novas funções ficaram mais simples?

6.0.2) Analise os exemplos e exercícios de números naturais e defina uma função `reduz-num` (semelhante a função `reduz` feita em sala) que abstraia o comportamento do template para funções com números naturais. Redefina as funções (que for possível) em termos de `reduz-num`. As novas funções ficaram mais simples?

6.0.3) [sicp 1.34] Dado a seguinte definição

```
(define (f g)
  (g 2))
```

Então nós temos

```
> (f sqr)
4

> (f (λ (z) (* z (z + 1))))
6
```

O que acontecerá se nós (perversamente) pedirmos ao interpretador para avaliar a combinação `(f f)`. Explique.

6.0.4) [tspl 2.4.1] Reescreva as seguintes expressões usando `let` para remover as subexpressões em comum e melhorar a estrutura do código. Não faça nenhuma simplificação algébrica.

```
(+ (- (* 3 a) b) (+ (* 3 a) b))

(cons (first (list a b c)) (rest (list a b c)))
```

6.0.5) [tspl 2.4.3] Determine o valor da seguinte expressão. Explique como você chegou neste valor.

```
(let ([x 9])
  (* x
     (let ([x (/ x 3)])
       (+ x x))))
```

6.0.6) [tspl 2.5.1] Determine o valor das seguintes expressões

```
(let ([f (λ (x) x)])
  (f 4))

(let ([f (λ x x)])
  (f 4))

(let ([f (λ (x . y) x)])
  (f 4))

(let ([f (λ (x . y) y)])
  (f 4))
```

6.0.7) [tspl 2.5.2] Como o procedimento primitivo `list` pode ser definido?

6.0.8) Defina uma função que receba como parâmetro um predicado (função de um argumento que retorna um valor booleano) e uma lista, e conte quantos elementos da lista satisfazem o predicado. Exemplo

```
> (cont positive? (list 1 -1 2 3 -2 5))
4
```

6.0.9) [sicp 1.41] Defina a função `double` que receba como parâmetro uma função de um parâmetro e retorne uma função que aplique a função original duas vezes. Por exemplo, dado que a função `add1` adiciona 1 ao seu parâmetro, então `(double add1)` retorna uma função que adiciona 2 ao parâmetro. Qual é o valor retornado por

```
(((double (double double)) add1) 5)
```

6.0.10) Defina uma função `concatena` que receba como parâmetro um número variável de listas e calcule a concatenação de todos os parâmetros. Exemplo

```
> (concatena (list 1 2 3) (list 4) (list 5 6))
'(1 2 3 4 5 6)
```

6.0.11) A função `map` pré definida no Racket aceita como parâmetro uma função de aridade n e n listas do mesmo tamanho, e aplica a função a todos os primeiros elementos das listas, depois aplica a função a todos os segundos elementos das listas e assim por diante, retornando a lista de resultados. Defina a função `mapeia` que funciona como a função `map` pré-definida Exemplo

```
> (mapeia + (list 1 2 3) (list 4 5 6) (list 7 8 9))
'(12 15 18)
> (mapeia list (list 1 2 3) (list 4 5 6) (list 7 8 9))
'((1 4 7) (2 5 8) (3 6 9))
```

6.0.12) [sicp 2.20] Defina uma função que receba um ou mais inteiros como parâmetro e retorne uma lista com os parâmetros que tenha a mesma paridade do primeiro argumento. Exemplo

```
> (same-parity 1 2 3 4 5 6 7)
'(1 3 5 7)

> (same-parity 2 3 4 5 6 7)
'(2 4 6)
```

6.0.13) [sicp 2.4] A seguir é apresentado uma representação procedural para um par. Para esta representação, verifique que `(car (cons x y))` produz `x` para qualquer objeto `x` e `y`.

```
(define (cons x y)
  (λ (m) (m x y)))

(define (car z)
  (z (λ (p q) p)))
```

Qual é a definição correspondente de `cdr`? (Dica: para verificar que isto funciona, faça uso do modelo de substituição).

Referências

- [sicp]. Structure and Interpretation of Computer Programs
- [tspl]. The Scheme Programming Languague

Licença