Programação Funcional
Fundamentos

Marco A L Barbosa malbarbo.pro.br

Começando

1) O que é um literal?
2) O que é uma função primitiva?
3) O que é uma expressão?
4) O que significa avaliar uma expressão?
5) Qual é a regra de avaliação para uma chamada de função?
6) Qual é a regra de avaliação para uma expressão `case`?
7) Qual é o propósito de uma definição?
8) O que é uma função?
9) A ordem que as expressões em uma chamada de função são avaliadas pode alterar o valor da chamada da função? Explique.
10) O que é uma definição com autorreferência?
11) O que é um processo recursivo?

Praticando

12) Faça uma função chamada `area_retangulo` que recebe dois argumentos, a `largura` e a `altura` de um retângulo, e calcula a sua área. Use o método de substituição para verificar se a sua função funciona corretamente de acordo com os exemplos a seguir. Em seguida confira as respostas no modo interativo.

```
> area_retangulo(3.0, 5.0)
15.0
> area_retangulo(2.0, 2.5)
5.0
```

13) Faça uma função chamada `produto_anterior_posterior` que recebe um número inteiro `n` e calcula o produto de `n`, `n + 1` e `n - 1`. Use o método de substituição para verificar se a sua função funciona corretamente de acordo com os exemplos a seguir. Em seguida confira as respostas no modo interativo.

```
> produto_anterior_posterior(3)
24
> produto_anterior_posterior(1)
0
> produto_anterior_posterior(-2)
-6
```

14) Faça uma função chamada `so_primeira_maiuscula` que recebe uma palavra não vazia (string) como parâmetro e crie uma nova string convertendo a primeira letra da palavra para maiúscula e o restante da palavra para minúscula. Use o método de substituição para verificar se a sua função funciona corretamente de acordo com os exemplos a seguir. Em seguida confira as respostas no modo interativo.

```
> so_primeira_maiuscula("paula")
"Paula"
> so_primeira_maiuscula("ALFREDO")
"Alfredo"
```

15) Faça uma função chamada `eh_par` que recebe um número natural `n` e indica se `n` é par. Um número é par se o resto da divisão dele por 2 é igual a zero. Não use `case` e nem a função pré-definida `int.is_even`. Use o método de substituição para verificar se a sua função funciona corretamente de acordo com os exemplos a seguir. Em seguida confira as respostas no modo interativo.

```
> eh_par(3)
False
> eh_par(6)
True
```

16) Faça uma função chamada `tem_tres_digitos` que recebe um número natural `n` e verifica se `n` tem exatamente 3 dígitos. Não use `case`. Use o método de substituição para verificar se a sua função funciona corretamente de acordo com os exemplos a seguir. Em seguida confira as respostas no modo interativo.

```
> tem_tres_digitos(99)
False
> tem_tres_digitos(100)
True
> tem_tres_digitos(999)
True
> tem_tres_digitos(1000)
False
```

17) Faça uma função `maximo` que encontre o máximo entre dois inteiros. Não use a função `int.max`. Use o método de substituição para verificar se a sua função funciona corretamente de acordo com os exemplos a seguir. Use o método de substituição para verificar se a sua função funciona corretamente de acordo com os exemplos a seguir. Em seguida confira as respostas no modo interativo.

```
> maximo(3, 5)
5
> maximo(8, 4)
8
> maximo(6, 6)
6
```

18) Faça uma função chamada `ordem` que recebe três inteiros distintos, `a`, `b` e `c` e determina se a sequencia `a, b, c` está em ordem crescente, decrescente ou não está em ordem. Use os operadores relacionas com três argumentos. Use o método de substituição para verificar se a sua função funciona corretamente de acordo com os exemplos a seguir. Em seguida confira as respostas no modo interativo.

```
> ordem(3, 8, 12)
"crescente"
> ordem(3, 1, 4)
"sem ordem"
> ordem(3, 1, 0)
"decrescente"
```

Avançando

19) [sicp 1.4] O modelo de avaliação visto em sala permite combinações em que os operadores são expressões compostas. Use esta observação para descrever o comportamento do seguinte procedimento:

```
fn a_plus_abs_b(a, b) {
  case b > 0 {
    True -> int.add
    False -> int.subtract
  }(a, b)
}
```

20) [sicp 1.5] Ben Bitdiddle inventou um método para determinar se um interpretador está usando avaliação com ordem aplicativa ou avaliação com ordem normal. Ele definiu os seguintes procedimentos:

```
fn p() {
  p()
}

fn test(x, y) {
```

```
  case x == 0 {
      True -> 0
      False -> y
  }
}
```

Então avaliou a seguinte expressão

```
test(0, p())
```

Qual é o comportamento que Ben irá observar com um interpretador que usa avaliação com ordem aplicativa? Qual é o comportamento que ele irá observar com um interpretador que usa avaliação com ordem normal? Explique a sua resposta.