Princípios de Programação

Exercícios

Universidade de Lisboa
Faculdade de Ciências
Departamento de Informática
Licenciatura em Engenharia Informática

2019/2020

A sintaxe das funções

Tópicos endereçados neste capítulo: *Pattern matching*, guardas em funções, as expressões `where` e `let-in`.

1. Usando *pattern matching* escreva funções que devolvam:

   (a) O primeiro elemento de um par

   (b) Um dado par com a ordem dos elementos trocados

   (c) O primeiro elemento de um triplo

   (d) Um dado triplo com os dois primeiros elementos trocados

   (e) O segundo elemento de uma lista

   (f) O segundo elemento do primeiro par de uma lista de pares

   Poderia fazê-lo sem *pattern matching*?

2. Defina a função `somaVec :: (Double,Double)-> (Double, Double)-> (Double,Double)` que soma dois vetores no plano representados por pares. Utilize *pattern matching*.

3. As funções abaixo diferem? Se sim, como?

   (a)
   ```
   hd1 (x:_)= x
   ```

   (b)
   ```
   hd2 :: [Int] -> Int
   hd2 (x:_)= x
   ```

   (c)
   ```
   hd3 :: [a] -> a
   hd3 (x:_)= x
   ```

4. Qual a diferença entre as seguintes funções?

   (a)
   ```
   f1 0 = 0
   f1 x = x - 1
   ```

   (b)
   ```
   f2 x = if x == 0 then 0 else x - 1
   ```

   (c)
   ```
   f3 x = x - 1
   f3 0 = 0
   ```

   (d)
   ```
   f4 x | x /= 0    = x - 1
        | otherwise = 0
   ```

5. Defina uma função que receba um par representando a componente real e imaginária de um número complexo. Essa função deverá devolver o quadrante em que esse ponto se encontra.

6. Escreva uma função que devolva a abreviatura do ordinal em inglês para um número cardinal positivo. Ex. `ordinalPrefix 31` deverá devolver `"31st"` e `ordinalPrefix 7` deverá devolver `"7th"`.

7. Implemente a função `leetSpeak` que receba uma string em minúsculas e devolva essa string convertida segundo algumas regras:

   - a letra `a` é substituída por um 4;
   - a letra `i` é substituída por um 1;
   - a letra `t` é substituída por um 7;
   - a letra `o` é substituída por um 0;
   - a letra `s` é substituída por um 5;
   - a letra `e` é substituída por um 3;
   - e todas as restantes letras deverão ser convertidas para maiúsculas.

   Como exemplo, `leetSpeak "i␣am␣totally␣so␣very␣leet"` deverá retornar `"1␣4M␣7O74LLY␣5O␣V3RY␣L337"`.

8. Considere a função `safetail` que se comporta como **`tail`**, excepto que transforma a lista vazia na lista vazia. Defina `safetail` utilizando:

   (a) uma expressão condicional,

   (b) equações guardadas,

   (c) *pattern matching*.

9. Utilizando as funções sobre listas constantes no `prelude`, escreva a função `halve :: [a] -> ([a],[a])` que separa uma lista em duas sublistas com comprimentos que não difiram de mais de uma unidade. Por exemplo:

```
ghci> halve [1..6]
([1,2,3],[4,5,6])
```

(a) Utilize o idioma **where**

(b) Utilize a expressão **let-in**

10. Escreva uma função que devolva o par de raízes de um polinómio do segundo grau, assumindo que o polinómio tem raízes reais. Dado um polinómio da forma $ax^2 + bx + c$, a função recebe três parâmetros a, b e c. Utilize o idioma **where**.

11. Escreva uma variante da função do exercício anterior em que a função devolve uma lista com as várias raízes reais do polinómio da seguinte forma: lista vazia se o polinómio não tiver raízes reais, lista com um elemento se o polinómio tiver uma raiz real (de multiplicidade dois), lista com dois elementos se o polinómio tiver duas raízes reais distintas. Utilize uma função com guardas.

12. Mostre como pode definir a disjunção lógica de três modos diferentes, utilizando *pattern matching*. Defina a disjunção como um operador infixo `\/`. Compare as definições avaliando **True** `\/` **undefined** e **False** `\/` **undefined**. Nota: **undefined** é uma constante pré-definida que representa uma computação divergente (que não termina). A constante polimórfica **undefined** redunda num erro quando a tentamos avaliar.

13. Encontre o valor e um tipo para a expressão

```
zip xs ys
where xs = tail [0,1,2,3]
      ys = init ['a','b','c','d']
```