EXERCÍCIOS EM HASKELL

CONDICIONAIS E RECURSÃO

→ Exercícios para praticar funções

1) Crie um módulo chamado de funções diversas e salve em um script. Em seguida, declare as funções abaixo. Lembre-se de adicionar a assinatura de tipo para cada função. Após a implementação de cada função, teste o seu script usando o ambiente interativo GHCi:
   i. Carregar o script: **`:l <nome do script>`**
   ii. Testar cada uma das funções: **`<nome da funcao> <argumentos>`**

Importante: ao criar as funções, use a convenção recomendada para nomes de função e nomes de parâmetro.

   a) Função para calcular a soma dos quadrados: recebe três valores numéricos e retorna a soma dos quadrados dos três valores. Faça três implementações: a primeira sem usar funções prontas, a segunda usando a função ** do módulo Prelude e a terceira usando a função ^ do módulo Prelude. Pesquise a diferente entre as funções ** e ^.

   b) Função para verificar se um número ímpar: recebe um número inteiro e retorna se ele é ímpar ou não. Faça duas implementações, uma usando a função *even* do módulo Prelude e outra usando a função *mod*.

→ Exercícios para praticar condicionais

2) Crie o módulo testando condicionais e declare as funções abaixo. Em seguida, teste as funções que você criou usando o ambiente interativo GHCi.

   a. Escreva uma função que receba a idade de uma pessoa e determine se ela possui idade para consumir bebidas alcoólicas. Considere que a idade mínima permitida para consumo de álcool seja 18. A função deve retornar uma string.
      - Implemente a versão com guardas
      - Implemente a versão com if-then-else

   b. Escreva uma função que receba um número e determine se ele é positivo, negativo ou igual a zero. A função deve retornar uma string.
      - Implemente a versão com guardas
      - Implemente a versão com if-then-else

   c. Escreva uma função que verifique se um ano é bissexto ou não. A função deve retornar true ou false.
      Dica: Um ano é bissexto se for divisível por 4, exceto quando é divisível por 100. No entanto, se um ano for divisível por 400, ele também é bissexto.
      - Implemente a versão com if-then-else
      - Implemente a versão com if-then-else usando where
      - Implemente a versão com guardas usando where

→ Exercícios para praticar função recursiva

3) Crie um módulo chamado recursão e declare as seguintes funções recursivas:

   a. Escreva uma função recursiva que gere o n-ésimo termo da sequência de Fibonacci usando recursão. Dica: a função retorna 0 quando n é igual a 0; 1 quando n é igual a 1; e a soma dos dois termos anteriores do n-éstimo termo.

   b. Escreva uma função recursiva que conte o número de dígitos de um número inteiro. Considere que o número digitado como entrada é sempre maior que zero.

   c. Escreva uma função recursiva que calcule a soma dos dígitos de um número inteiro. Considere que o número digitado como entrada é sempre maior que zero.

→ Desafio

1) Crie um módulo chamado "primeiro programa" dividido em três partes. Lembre-se de adicionar comentários a cada parte do seu código.

   a. Base de dados

      Nessa parte, declare 5 funções para representar os números reais de uma sequência. Por exemplo:

```
num 1 = 5.0
num 2 = 10.0
...
```

(Lê-se: a função *num* quando recebe *1* como entrada retorna *5.0*)
(Lê-se: a função *num* quando recebe *2* como entrada retorna *10.0*)

   b. Função recursiva (soma)

      Implemente uma função recursiva para somar todos os números retornados pelas funções da base de dados (num 1 + num 2 + ... num 5). A função, de preferência, não deve usar if-then-else ou guardas.

   c. Função (media)

      Implemente uma função (não recursiva) para calcular a média dos números retornados pelas funções da base de dados. Dica: a função ***fromIntegral*** converte um inteiro para real e possui a seguinte assinatura de tipo:

```
fromIntegral :: Int -> Float
```

Em seguida, execute o programa que você criou usando o ambiente interativo GHCi. Por exemplo:

```
> :l CalculoMedia
> media 5
```