| | |
|---|---|
| CURSO: Ciência da Computação<br>PERÍODO: 4o.<br>DISCIPLINA: Técnicas Alternativas de Programação | CÓDIGO: CI062<br>PROFESSOR: Andrey<br>AULA: 22 |

# 1 Apresentação

Na aula de hoje vamos apresentar e discutir e exercitar a implementação de uma árvore binária em Haskell, usando o conceito de tipos estruturados. A aula de hoje é baseada em [de Sá and da Silva, 2006]

# 2 Desenvolvimento

## 2.1 Tipos Estruturados

A criação de novos tipos é feita pelo pelo construtor **data**. Este tipo de construção permite a criação de tipos mais estruturados como enumerações e tipos com variações ou recursivos. Por exemplo:

```
data Cor = Verde | Azul | Amarelo
   deriving (Eq, Show)

corBasica :: Cor -> Bool

corBasica c = (c == Verde || c == Azul || c == Amarelo)
```

No exemplo, a expressão **deriving (Eq, Show)** significa que este novo tipo "herda"as propriedades das classes Eq e Show. A classe Eq permite a comparação de igualdade entre elementos deste tipo e a Show permite mostrar elementos deste tipo no terminal.

## 2.2 Árvore Binária

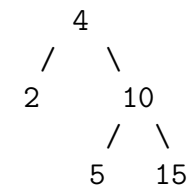
Um outro exemplo é a definição dos tipos necessários para uma árvore Binária. Este tipo de estrutura deve possuir uma definição recursiva de tipos. Um definição recursiva também pode ser usada para implementar uma lista encadeada. Abaixo temos uma definição de tipos para uma árvore binária.

```
data ArvoreBin = NodoNull | NodoInt Int ArvoreBin ArvoreBin
      deriving Show
```

O tipo ArvoreBin pode ser um NodoNull, que é um símbolo que indica um nó nulo (ponteiro nulo em C) ou um nó completo (Nodo Int ArvoreBin ArvoreBin) onde o símbolo Nodo serve para indicar que é um nó completo, o elemento que é um Int e os dois campos que apontam para outros nós.

```
arvoreBin = NodoInt 4 (NodoInt 2 Nodonull Nodonull)
                      (NodoInt 10 (NodoInt 5 Nodonull Nodonull)
                                  (NodoInt 15 Nodonull Nodonull))
```

A função arvoreBin retorna uma árvore básica já construída.




Como primeiro exercício, vamos implementar um passeio pela árvore. Para facilitar, vamos representar o passeio como uma lista e a cada nó que visitamos é criada uma lista com o elemento que é então concatenada com as listas dos nós visitados recursivamente.

```
passeio :: ArvoreBin -> [Int]

passeio Nodonull = []

passeio (NodoInt a Nodonull Nodonull) = [a]

passeio (NodoInt a esq dir) = [a] ++ passeio esq ++ passeio dir
```

O resultado da chamada de função passeio é:

```
*Main> passeio arvoreBin
[4,2,10,5,15]
```

## 2.3 Inserção em uma Árvore Binária de Busca

O próximo exercício é implementar a inserção de um elemento na árvore, inserindo-o como uma folha seguindo as restrições de uma árvore binária de busca. Primeiro vamos criar uma função que cria um novo nó a partir de um elemento recebido como parâmetro.

```
criaNodo :: Int -> ArvoreBin

criaNodo a = (NodoInt a Nodonull Nodonull)
```

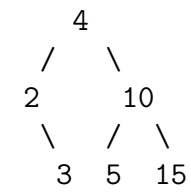
A seguir, vamos implementar a função de inserção de um elemento na árvore, que precorre a árvore na sua altura até achar o lugar onde será inserido o elemento, então chamando a função criaNodo.

```
insere :: Int -> ArvoreBin -> ArvoreBin

insere x Nodonull = criaNodo x

insere x (NodoInt a esq dir)
     |x < a = (NodoInt a (insere x esq) dir)
     |x > a = (NodoInt a esq (insere x dir))
```

Chamando a função passeio para mostrar a inserção do elemento 3 na árvore criada pela função arvoreBin temos.

```
*Main> passeio (insere 3 arvoreBin)
[4,2,3,10,5,15]
```

O que representa a seguinte árvore já com o elemento 3 inserido.




# 3 Atividades

**1:** Construa uma função realize uma rotação à direita em um nó da árvore.

**2:** Construa uma função realize uma rotação à esquerda em um nó da árvore.

**3:** Construa uma função realize uma rotação dupla à direita em um nó da árvore.

**4:** Construa uma função realize uma rotação dupla à esquerda em um nó da árvore.

**5:** Construa uma árvore binária de busca AVL, com sua definição e a função de inserção e alterando as funções acima para manter a condição de AVL.

# Referências

[de Sá and da Silva, 2006] de Sá, C. C. and da Silva, M. F. (2006). *Haskell*. Novatec Editora.