Programação Funcional

(COMP0393)

Leila M. A. Silva

dcomp
departamento de computação

Tipos Abstratos de Dados

(COMP0393)

Aula 16

dcomp
departamento de computação

Tipos Abstratos de Dados (TADs)

- Permite esconder informação na definição de tipos.
- Exporta apenas o necessário para usar o tipo.
- Úteis para implementar estruturas de dados.
- Exemplo: Filas. Simulam filas na vida real.
- Filas são estruturas que armazenam dados que possuem o seguinte mecanismo: o novo dado é inserido sempre no final da fila. O dado a ser imediatamente processado é aquele que está no início da fila.
- As operações clássicas em estruturas que armazenam dados são:
  - Defina a estrutura como vazia para iniciar o processo
  - Estrutura está vazia?
  - Insira na estrutura
  - Remova da estrutura

TADs

- Sintaxe: Para introduzir um TAD crie um módulo para o TAD

```
module Fila (Fila, filaVazia, ehVazia,
             insere, remove) where

newtype Fila a = Fil [a]

filaVazia :: Fila a
filaVazia = Fil []

ehVazia :: Fila a -> Bool
ehVazia (Fil []) = True
ehVazia _        = False
```

Idêntico a `data`, porém mais eficiente quando só tem um construtor.
Observe que o tipo é exportado pelo módulo

Define uma fila vazia

Checa se a fila é vazia

TADs

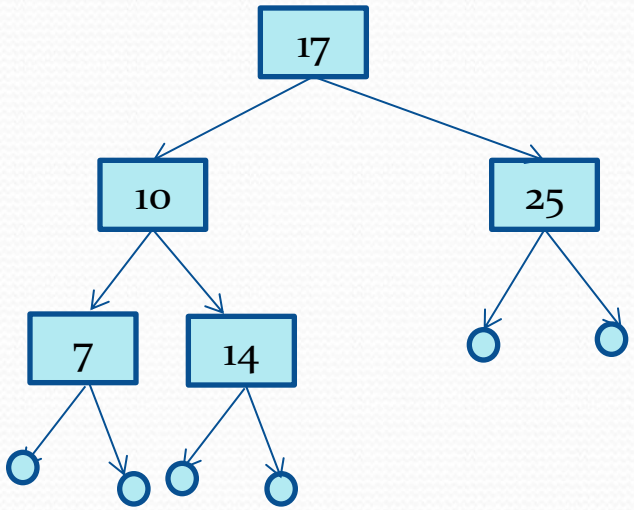
```
module Fila (Fila, filaVazia, ehVazia,
             insere, remove) where

-- continuando o módulo Fila

insere :: a -> Fila a -> Fila a
insere  x (Fil xs) = Fil (xs ++ [x])

remove :: Fila a -> (a, Fila a)
remove q@(Fil xs)
   | not (ehVazia q) = (head xs, Fil (tail xs))
   | otherwise       = error " nao ha elementos a remover"
```

Insere na fila

Remove da fila

Mais exemplos de TADs

- Árvore binária de busca: árvore binária em que todos os nós à esquerda de um dado nó são menores que ele e os nós à direita de um dado nó são maiores que ele
- Algumas operações clássicas são:
  - Defina uma árvore como vazia
  - A árvore está vazia?
  - Verifique se um nó é nó nulo
  - Colete a subárvore esquerda
  - Colete a subárvore direita
  - Colete a informação de um nó não nulo
  - Insira na árvore
  - Remova da árvore



Mais exemplos de TADs

```
module ABB (Arv, arvVazia, ehVazia,
            insere, remove, ehNoNulo,
            arvEsq, arvDir, infoNo) where

data Arv a = NoNulo | No a (Arv a) (Arv a)

arvVazia :: Arv a
arvVazia = NoNulo

ehVazia :: Arv a -> Bool
ehVazia (NoNulo) = True
ehVazia _        = False
```

Define uma árvore vazia

Checa se a árvore é vazia

Mais exemplos de TADs

```
module ABB (Arv, arvVazia, ehVazia,
            ehNoNulo, arvEsq, arvDir,
            infoNo, insere, remove, ) where

-- continuando o TAD

ehNoNulo :: Arv a -> Bool
ehNoNulo NoNulo = True
ehNoNulo _      = False

arvEsq :: Arv a -> Arv a
arvEsq (NoNulo) = error " nao tem subarvore esquerda"
arvEsq (No _ tesq _) = tesq

arvDir :: Arv a -> Arv a
arvDir (NoNulo) = error " nao tem subarvore direita"
arvDir (No _ _ tdir) = tdir
```

Checa se um nó é nulo

Coleta subárvore esquerda

Coleta subárvore direita

Mais exemplos de TADs

```
module ABB (Arv, arvVazia, ehVazia,
            ehNoNulo, arvEsq, arvDir,
            infoNo, insere, remove, ) where
-- continuando o TAD
ehNoNulo :: Arv a -> Bool
ehNoNulo NoNulo = True
ehNoNulo _      = False

arvEsq :: Arv a -> Arv a
arvEsq (NoNulo) = error " nao tem subarvore esquerda"
arvEsq (No _ tesq _) = tesq

arvDir :: Arv a -> Arv a
arvDir (NoNulo) = error " nao tem subarvore direita"
arvDir (No _ _ tdir) = tdir

infoNo :: Arv a -> a
infoNo NoNulo = error "arvore vazia"
infoNo ( No x _ _ ) = x
```

Checa se um nó é nulo

Coleta subárvore esquerda

Coleta subárvore direita

Coleta informação do nó

Mais exemplos de TADs

```
module ABB (Arv, arvVazia, ehVazia,
            ehNoNulo, arvEsq, arvDir,
            infoNo, insere, remove, ) where

-- continuando o TAD

insere :: Ord a => a -> Arv a -> Arv a
insere  x NoNulo = (No x NoNulo NoNulo)
insere x (No y tesq tdir)
  | x == y = (No x tesq tdir)
  | x > y = No y tesq (insere x tdir)
  | otherwise = No y (insere x tesq) tdir
```

insere na árvore

O elemento já existe e a árvore não se altera

Insere recursivamente na subárvore direita ou esquerda

Mais exemplos de TADs

```
module ABB (Arv, arvVazia, ehVazia,
            ehNoNulo, arvEsq, arvDir,
            infoNo, insere, remove, ) where

-- continuando o TAD

remove :: Ord a => a -> Arv a -> Arv a
remove  x NoNulo = error " nao há elementos a remover"
remove x (No y tesq tdir)
  | x < y = No y (remove x tesq) tdir
  | x > y = No y tesq (remove x tdir)
  | ehNoNulo tdir = tesq
  | ehNonulo tesq = tdir
  | otherwise = una tesq tdir
```

remove na árvore

Remove recursivamente na subárvore esquerda ou direita

x=y, se um dos filhos do nó removido é nulo, o outro filho assume o lugar do nó removido

x=y, mas os dois filhos não são nulos preciso localizar o elemento certo para assumir o lugar do pai sem modificar a ordem da ABB (função `una`)

Mais exemplos de TADs

```
module ABB (Arv, arvVazia, ehVazia,
            ehNoNulo, arvEsq, arvDir,
            infoNo, insere, remove, ) where
-- continuando o TAD
una :: Ord a => Arv a -> Arv a -> Arv a
una tesq tdir = No mini tesq novaArv
  where (Just mini) = minArv tdir
        novaArv = remove mini tdir

minArv :: Ord a => Arv a -> Maybe a
minArv t
  | ehNoNulo t = Nothing
  | ehNoNulo (arvEsq t) = Just (infoNo t)
  | otherwise = minTree (arvEsq t)
```

Busca sucessor do nó removido na árvore

```
data Maybe a = Nothing | Just a                -- error type
               deriving (Eq, Ord, Show, Read)
```

Usando o TAD

```
import ABB

tamanho :: Arv a -> Int
tamanho t
  | ehNonulo t = 0
  | otherwise = 1 + tamanho (arvEsq t) + tamanho (arvDir t)
```

Observe que se resolvermos mudar a implementação do módulo desde que a interface se mantenha esta função permanece inalterada

Exercícios Recomendados

- Defina o TAD `Pilha` que implementa o seguinte mecanismo: um dado novo é inserido sempre no início da pilha e removido do seu início também, similar a uma pilha de pratos na pia. Ao resolver a questão defina funções similares a do tipo `Fila`.
- Defina o TAD `Database` para armazenar o banco de dados do exemplo da biblioteca já visto em sala de aula. Insira no TAD operações de remoção, inserção e consulta de itens pelos dois atributos de nome e livros e verificação de o banco de dados é vazio.
- Modifique a implementação do TAD `ABB` para incluir no nó a informação de quantas vezes um elemento aparece na árvore. Inclua a função que dado um valor devolve quantas vezes ele ocorre na árvore.