Aula Prática 7

Paradigmas da Programação I / Programação Funcional

ESI/MCC 1^o ano (2005/2006)

1 Árvores de Expressão

Considere os seguintes tipos de dados utilizados para a representação de expressões aritméticas por árvores binárias:

```
data OP = SOMA | SUB | PROD | DIV
   deriving (Show, Eq)
data Expr = Folha Int | Nodo OP Expr Expr
   deriving (Show, Eq)
```

Tarefa 1

1. Escreva uma função `aplica` que aplica um operador binário a dois argumentos inteiros:

   ```
   aplica :: OP -> Int -> Int -> Int
   ```

2. Escreva uma função `avalia` que procede ao cálculo do valor de uma expressão:

   ```
   avalia :: Expr -> Int
   ```

2 Árvores Binárias de Pesquisa

Considere que se pretende utilizar uma árvore binária de pesquisa para guardar informação sobre os alunos inscritos nesta disciplina. Para cada aluno deve-se guardar o nome, número, nota de avaliação prática e nota de avaliação teórica.

Relembre que estas árvores se caracterizam pela seguinte propriedade (invariante de ordem): o conteúdo de qualquer nó situado à esquerda de um nó X é necessariamente menor do que o conteúdo de X, que por sua vez é necessariamente menor do que o conteúdo de qualquer nó situado à sua direita. Admitindo-se a ocorrência de elementos iguais, uma destas restrições é relaxada para "menor ou igual".

Tarefa 2

1. Defina tipos de dados `Aluno` e `ArvBinAluno` apropriados. Declare também constantes destes tipos.

2. Escreva funções de inserção e de pesquisa apropriadas para a manipulação de árvores de pesquisa de alunos, considerando que a ordenação é feita *por número*. Note que cada aluno tem um número único.

   ```
   insereNumAluno :: Aluno -> ArvBinAluno -> ArvBinAluno
   pesquisaNumAluno :: Int -> ArvBinAluno -> Maybe Aluno
   ```

3. Defina uma função que efectue uma travessia `inorder` de uma árvore de pesquisa:

   ```
   inorder :: ArvBinAluno -> [Aluno]
   ```

Tarefa 3

Considere agora que se pretende trabalhar simultaneamente com duas árvores: além da árvore inicial ordenada por número de aluno, pretende-se ter também uma segunda árvore, ordenada por nome de aluno (alfabéticamente).

1. Escreva funções de inserção e pesquisa apropriadas. A chave da pesquisa deve ser o nome do aluno. Admita que podem existir vários alunos com o mesmo nome.

   ```
   insereNomeAluno :: Aluno -> ArvBinAluno -> ArvBinAluno
   pesquisaNomeAluno :: String -> ArvBinAluno -> [Aluno]
   ```

2. Escreva uma função

   ```
   converteNomeNum :: ArvBinAluno -> ArvBinAluno
   ```

   que produz uma árvore ordenada por número a partir de uma árvore ordenada por nome. O que terá que alterar nesta função para efectuar a conversão em sentido contrário?

3. Utilizando funções já definidas ao longo desta ficha, escreva duas funções de ordenação de listas de aluno que utilizem como critério o nome e o número, respectivamente.

   ```
   sortNomeAluno :: [Aluno] -> [Aluno]
   sortNumAluno :: [Aluno] -> [Aluno]
   ```