Paradigmas da Programação I (5301P3)

LESI 2004/05

22 de Setembro de 2004

Objectivos

Na actual estrutura da Licenciatura em Engenharia de Sistemas e Informática, esta é a primeira disciplina de Informática. Por esta razão, o primeiro grande objectivo desta disciplina é a introdução à programação de computadores.

Dos vários paradigmas de programação (imperativo, funcional, lógico e orientado a objectos) apresentados ao longo da licenciatura, nesta disciplina estuda-se o paradigma funcional, tendo por base a linguagem de programação Haskell. Esta abordagem tem a vantagem de não exigir dos alunos grandes conhecimentos prévios sobre a arquitectura física dos computadores, para poderem entender e escrever programas funcionais, permitindo, por esse motivo, trabalhar mais facilmente com turmas heterogéneas em conhecimentos de informática.

A linguagem Haskell é uma linguagem puramente funcional fortemente tipada e com um sistema de tipos extremamente rico, o que permite não só ensinar os conceitos fundamentais da programação (funcional), onde se incluem tópicos tais como: tipos de dados, estruturas de controlo, e recursividade; como também ensinar conceitos avançados, como por exemplo: funções de ordem superior, polimorfismo, classes, modularidade, e monades.

Programa Detalhado

1. Introdução ao paradigma funcional de programação.
2. Conceitos fundamentais da programação em Haskell.
   (a) Noção de programa e a sua execução.
   (b) Expressões e valores.
   (c) Redução.
   (d) Tipos.
   (e) Definições.
   (f) Estruturas condicionais e concordância de padrões.
   (g) Definições locais.

(h) Módulos.

3. Funções.
   (a) Funções pré-definidas sobre tipos básicos.
   (b) Funções não recursivas.
   (c) Funções recursivas.
   (d) Parâmetros de acumulação.
   (e) Polimorfismo.
   (f) Funções anónimas.
   (g) Funções de ordem superior.
4. Listas.
   (a) O conceito de lista.
   (b) Listas por compreensão.
   (c) Listas infinitas.
   (d) Operações básicas sobre listas.
   (e) As funções `map` e `filter`.
   (f) As funções `foldr` e `foldl`.
   (g) Outras funções de ordem superior.
   (h) Funções para manipulação de texto.
   (i) Algoritmos de ordenação.
      i. Insertion sort.
      ii. Merge sort.
      iii. Quick sort.
5. Tipos, classes e polimorfismo.
   (a) Definição de novos tipos de dados.
   (b) Classes e instâncias.
   (c) Classes derivadas.
   (d) Hierarquia de classes.
   (e) Algumas classes pré-definidas.
   (f) Polimorfismo e sobrecarga.
   (g) Classes de construtores.
6. Árvores.
   (a) Árvores binárias.
   (b) Árvores binárias de procura.

(c) Árvores generalizadas.

7. Monades.
   (a) Noção de monade e sua definição.
   (b) O monade `IO`.
   (c) O monade `Maybe`.
   (d) Outros monades.
8. Tipos Abstractos de Dados
   (a) O conceito de tipo abstracto de dados e a sua implementação em Haskell.
   (b) Alguns exemplos: conjuntos, tabelas, pilhas e filas de espera.

Método de Avaliação

A avaliação tem uma componente teórica e uma componente prática, ambas obrigatórias. A nota final será calculada com base na seguinte fórmula:

$$\text{Nota Final} = \mathbf{nt} \times 0.6 + \mathbf{np} \times 0.4$$

sendo

nt a nota teórica (**nt** ≥ 9 obrigatóriamente), obtida através da realização de uma prova individual escrita;

np a nota prática (**np** ≥ 10 obrigatóriamente), resultante da avaliação obtida ao longo das aulas laboratoriais (em regime de avaliação contínua) e que terá por base a realização de fichas de trabalho e trabalhos práticos.

Bibliografia

- Fundamentos da Computação, Livro II: Programação Funcional. José Manuel Valença e José Bernardo Barros. Universidade Aberta, 1999.
- Introduction to Functional Programming using Haskell. Richard Bird. Prentice-Hall, 1998.
- Haskell: the craft of functional programming. Simon Thompson. Addison-Wesley.
- Introduction to Functional Programming. Richard Bird and Philip Wadler. Prentice-Hall, 1988.
- A Gentle Introduction to Haskell. Paul Hudak, John Peterson and Joseph Fasel. (http://www.haskell.org/tutorial/)