Introdução à Unidade Curricular

Paradigmas de Programação

Materiais e Atendimento

Alexandra Mendes
Gabinete: 3.19
Email: amendes@di.ubi.pt
Materiais: Moodle

Atendimento: Quartas, 11h-13h
Nota: Requer marcação prévia por email com 48 horas de antecedência.

Funcionamento da UC

Semanal:

- 2 horas teóricas (com alguns exercicíos práticos)

  Sala: 6.18

- 2 horas práticas (com alguma teoria)

  Sala: 6.20

Quem tiver computador portátil é aconselhado a trazer para as aulas teóricas e práticas.

Detalhes da Avaliação

Avaliação em Frequência:

Teste 1 (T1): 31 de Março 2020, na aula prática (14h-16h)

Teste 2 (T2): 19 de Maio 2020, na aula prática (14h-16h)?

Nota dos testes vale 70%: (T1+T2)/2

Trabalho Prático: Entrega : 1 de Junho; Apresentação: 2 de Junho durante as durante as aulas no dia 2 de Junho?

Trabalho prático vale 30%.

Avaliação em Exame: Nota do exame é a nota final
Data será anúnciada mais tarde website dos Serviços Académicos

Mínimos para ser aprovado à disciplina:

Por Frequência >= 9.5 valores

Por Exame >= 9.5 valores

Mínimos para ser admitido a exame:

Média dos testes >= 7 valores
Nota do Trabalho Prático >= 7 valores

Conduta

- É esperado que venham às aulas e que se apliquem durante todo o tempo de aulas.
- É esperado que trabalhem para a UC fora do tempo de aulas e que explorem conceitos além dos apresentados na sala.
- Plágio de qualquer tipo implicará reprovação de todos os envolvidos (i.e. quem copiou e quem deixou copiar)

Respeitem toda a gente na sala de aula!

Folha de presenças vai circular nas teóricas e práticas.

O que é que devem fazer?

- Coloquem questões (estejam à vontade para interromper a aula se precisarem de alguma clarificação).
- Sejam interactivos e activos nas aulas (Learning is not a spectator sport!).
- Assumam a responsabilidade pela vossa aprendizagem. Estamos no ensino superior!
- Resolvam todos os exercicíos colocados (teóricas e práticas) e mais!
- Façam trabalho extra: leiam os livros recomendados, procurem mais informação e desafios online, etc
- **Ajudem-se uns ao outros!**

Está alguma coisa a incomodar-te? Estás a ter dificuldades?
Então entra em contacto comigo!

Bibliografia
(alguns livros recomendados)

- **“Programming in Haskell”, 2nd Edition, Graham Hutton, Cambridge University Press, 2016**
- “Introduction to Functional Programming Using Haskell”, Richard Bird, Prentice Hall Press, 1998
- “Real World Haskell”, Bryan O’Sullivan, Don Stewart, e John Gerzen, O’Reilly, 2008
- “Types and Programming Languages”, Benjamin Pierce, The MIT Press.
- “The Implemementation of Functional Programming Languages”, Simon Peyton Jones, PrenticeHall International.
- “Foundation for Programming Languages”, J. Mitchell, Foundations of Computing, MIT Press, 1996.
- “Semantics with Applications An Appetizer”, H. R. Nielson and F. Nielson. Springer, 2007
- **E outros poderão ser sugeridos durante as aulas. Ver sempre a bibliografia no Moodle.**

Objectivos da UC

Objectivos Principais:

- Ser capaz de resolver de forma fluente problemas de programação utilizando o paradigma funcional, em particular na linguagem Haskell.
- Raciocinar sobre, e manipular algebricamente, programas escritos em Haskell.

Conteúdos Programáticos da UC

1. Introdução aos paradigmas da programação
2. Semânticas da programação: operacional, axiomática e denotacional
3. Variáveis, Funções e valores no paradigma funcional
4. Tipos estruturados, polimórficos, de ordem superior e sistemas de tipos
5. Aplicação aos algoritmos e às estruturas de dados
6. Avaliação preguiçosa
7. Princípios de compilação e de execução de programas funcionais
8. Princípios algébricos da programação funcional
9. Cálculo e transformação de programas

Ambiente de Programação

1. Haskell-platform

   https://www.haskell.org/platform/

2. A Haskell Tool Stack
   https://docs.haskellstack.org/en/stable/README/

3. VS Code com o plugin Simple GHC (Haskell) Integration (que também vai instalar o Haskell syntax highlighting)

Aconselho o uso de Linux.

Outros Tópicos

- Dúvidas?

Paradigmas de Programação

Imperativo: O programador indica como mudar o estado (i.e. “como fazer” e não o “que fazer”)

- **Procedural:** Baseada em procedimentos que contêm um conjunto de passos computacionais a serem executados (Exemplos: C, Pascal, Basic).
  Programa = algoritmos + dados
- **Orientada a objectos:** Agrupa as instruções com o estado sobre o qual estas operam, i.e. **objectos** que interagem entre si (Exemplos: Java, C++).
  Programa = objectos + mensagens

Declarativo: O programador declara propriedades do resultado desejado e não como o computar.

- **Lógica:** O resultado desejado é declarado como a resposta a uma questão sobre um sistema de factos e regras (Exemplos: Prolog, Answer Set Programming - ASP).
  Programa = factos + regras.
- **Funcional:** Baseado em funções. O resultado desejado é declarado como uma série de aplicações de funções. Computações são a avaliação de funções matemáticas e evita alteração de estado e dados mutáveis (Exemplos: Haskell, Ocaml).
  Programa: funções compostas com funções.

Programação Funcional

- Método básico de computação é a aplicação de funções a argumentos.

- Torna os programas mais fáceis de compreender através da eliminação de alterações de estado que não dependem do input das funções, o que é chamado de efeitos colaterais (side effects).

- **Porquê aprender?**
  https://www.youtube.com/watch?time_continue=24&v=Fi83KOAqWb4&feature=emb_logo

- **Haskell na industria:** Facebook, Barclays, Google, e muito outros...
  https://wiki.haskell.org/Haskell_in_industry

Haskell

Algumas características importantes do Haskell (existem outras):

- **Programas concisos:** Normalmente os programas são mais concisos devido ao estilo high-level da programação funcional mas também porque a sintaxe foi desenhada para ser concisa, com poucas keywords e indentação para indicar a estrutura dos programas

- **Inferência de tipos:** Os tipos são inferidos automaticamente (mas podem também ser declarados pelo programador). O sistema de tipos do Haskell é muito poderoso e ajuda a detectar erros antes da execução dos programas. Suporta também polimorfismo e overloading.

- **Lazy evaluation:** Nenhuma computação é executada enquanto não for necessária. Evita computações desnecessárias, encoraja programação modular usando estruturas intermediárias, e até programação com estruturas infinitas.

- **Purely Functional:** Todas as funções em Haskell são funções no sentido matemático (i.e. são puras). Recebem todos os seus inputs como argumentos e os seu outputs como valores. No entanto, muitos programas requerem “side effects” – o Haskell fornece uma framework uniforme para programar com “side effect” mantendo as funções puras.

- **Higher-Order functions:** Funções podem receber funções como argumentos.