UNIVERSIDADE FEDERAL DE OURO PRETO
DEPARTAMENTO DE COMPUTAÇÃO

PLANO DE ENSINO

Nome do Componente Curricular em português: Programação Funcional **Nome do Componente Curricular em inglês:** Functional Programming		**Código:** BCC222
Nome e sigla do departamento: Departamento de Computação (DECOM)		**Unidade acadêmica:** ICEB
Nome do docente: Rodrigo Geraldo Ribeiro		
Carga horária semestral: 60 horas	**Carga horária semanal teórica:** 2 horas/aula	**Carga horária semanal prática:** 2 horas/aula
Data de aprovação na assembleia departamental: 20/08/2021		

Ementa:

Características dos principais paradigmas de programação; princípios do paradigma de programação funcional; principais características de linguagens de programação funcional: recursão, abstração funcional, funções de ordem superior, tipos de dados algébricos, polimorfismo, inferência de tipos, avaliação estrita e avaliação lazy, sobrecarga; estudo de uma linguagem funcional moderna e desenvolvimento de programas nesta linguagem, enfocando aspectos de correção, modularidade e reuso de código.

Conteúdo Programático:

- Introdução
- Paradigmas de programação
- Primeiros passos em haskell
- Definindo funções
- Tipos de dados
- Expressão condicional
- Funções recursivas
- Tuplas, listas e polimorfismo paramétrico
- Casamento de padrão
- Programas interativos
- Ações de e/s recursivas
- Números aleatórios e argumentos da linha de comando
- Arquivos
- Expressão lambda
- Funções de ordem superior
- Tipos algébricos
- Classes de tipos
- Mônadas

- Avaliação lazy
- Prova de propriedades de programas

Objetivos:

Ao final do curso espera-se que os alunos possuam os seguintes conhecimentos e habilidades:
- Conhecimento das características fundamentais de linguagens funcionais modernas e noções básicas sobre o modelo de execução de programas nessas linguagens;
- Noções básicas sobre sistemas de tipos e inferência de tipos;
- Capacidade de distinção entre polimorfismo paramétrico, de sobrecarga e de inclusão;
- Entendimento dos diferentes mecanismos de avaliação em linguagens de programação;
- Capacidade para comparar características de linguagens de diferentes paradigmas;
- Habilidade para programar em uma linguagem funcional moderna;
- Entendimento sobre a implementação da noção de estado em linguagens funcionais;
- Noções de derivação de programas a partir da especificação e sobre prova de correção de programas em linguagens funcionais.

Metodologia:

Aulas assíncronas sobre o conteúdo teórico. Aulas síncronas para resolução de exercícios e atendimento à dúvidas usando a plataforma Google Meet.

Requisitos de Hardware / Software

- Computador pessoal.
- Haskell stack: https://docs.haskellstack.org/en/stable/README/
- Agda programming language: https://wiki.portal.chalmers.se/agda/pmwiki.php

Aulas Práticas: Em cada conteúdo teórico serão disponibilizados exercícios para entrega posterior por parte dos alunos. Além disso, serão realizadas lives para dúvidas e resolução de exercícios.

Atividades avaliativas:

As atividades avaliativas serão realizadas no ambiente virtual de aprendizagem (moodle) e consistem em três Provas Teóricas (PT) no valor de 10 pontos. A média final será dada por:

média = (Prova1 + Prova2 + Prova3) / 3

Provas Teóricas: os alunos receberão uma prova com questões referentes aos conteúdos estudados e terão um tempo determinado para submeter a folha de respostas utilizando a plataforma Moodle.

Exame Especial: O Exame Especial será uma prova que deverá ser resolvida individualmente. Cada aluno receberá uma prova com questões referentes aos conteúdos estudados e terão um tempo determinado para submeter a folha de respostas na plataforma.

Resolução CEPE 2880 de 05/2006: É assegurado a todo aluno regularmente matriculado com frequência mínima de setenta e cinco por cento e média inferior a seis, o direito de ser avaliado por Exame Especial.

Para ser aprovado o(a) aluno(a) precisa ter nota final igual a 6,0 (seis) pontos em uma escala de zero a dez e frequência mínima de setenta e cinco por cento.

Cronograma:

Aulas Conteúdo

1	Apresentação da disciplina: critérios de avaliação e ementa. Introdução à linguagem Haskell.
2	Definindo funções simples.
3	Tipos e verificação de tipos em Haskell
4	Definindo funções recursivas
5	Definindo tipos de dados em Haskell
6	Aula para dúvidas e resolução de exercícios
7	Funções de ordem superior
8	Estudos de caso: Cifra de César e serialização
9	Estudo de caso: funções de ordem superior sobre árvores binárias.
10	Revisão para avaliação 1.
11	Avaliação 1.
12	Correção da avaliação 1. Polimorfismo de sobrecarga e classes de tipos.
13	Estrutura de projetos. Testes baseados em propriedades.
14	Avaliação lazy.
15	I/O em Haskell. Estudo de caso: implementando jogos usando console.
16	Aula de dúvidas e resolução de exercícios.
17	Functores e Functores aplicativos.
18	Construção de uma biblioteca de Parsing. Estudo de caso: Parsing de arquivos CSV.
19	Aula para dúvidas e resolução de exercícios.
20	Avaliação 2.
21	Correção da avaliação 2. Introdução às mônadas.
22	Correção de programas.
23	Correção de programas. Introdução à linguagem Agda.
24	Introdução à linguagem Agda.
25	Aula para dúvidas e resolução de exercícios. Revisão para avaliação 3.
26	Avaliação 3.
27	Correção da avaliação 3.
28	Entrega dos resultados parciais
29	Exame especial.
30	Entrega de resultados finais.

Bibliografia Básica:

- Lipovaca, Miran - Learn you a Haskell for the great good! No Starch Press. Disponível gratuitamente on-line:http://learnyouahaskell.com
- O'Sullivan, Bryan; Stewart, Don ; Goerzen, John - Real World Haskell. Disponível gratuitamente on-line:http://book.realworldhaskell.org
- WADLER, Philip; KOKKE, Wen; SIEK, Jeremy. Programming Languages Foundations in Agda. Disponível gratuitamente on-line: https://plfa.github.io
- MARLOW, Simon. Parallel and Concurrent Programming in Haskell. Disponível gratuitamente on-line: https://simonmar.github.io/pages/pcph.html

Bibliografia Complementar:

- Jhala, Ranjit; Seidel, Eric ; Vazou, Niki - Programming with Refinement Types: An introduction to Liquid Haskell.Disponível gratuitamente on-line:http://ucsd-progsys.github.io/liquidhaskell-tutorial/

- BIRD, Richard. Introduction to Functional Programming using Haskell. 2. ed. London: Prentice Hall Press, 1998.
- MILNER, R.; TOFTE, Mads; HARPER, Robert. The definition of standard ML. Cambridge: MIT, 1997.
- ULLMAN, Jeffrey D. Elements of ML programming. New Jersey: Prentice-Hall. 1998.
- PAULSON, Laurence C. ML for the working programming. 2. ed. Cambridge: Cambridge University, 1996.
- PEYTON JONES, Simon L. The implementation of functional programming languages. New York: Prentice-Hall. 1987.