MCTA016-13 - Paradigmas de Programação (Teoria e Prática)
Plano de ensino

Docentes: Diogo S. Martins e Emilio Francesquini
Centro de Matemática, Computação e Cognição
Universidade Federal do ABC

Q2 2018
v. 05/06

1 Informações básicas

- Aulas teóricas:
  - Turmas A1N e A2N:
    quintas 19-21h A106-0 semanal
- Aulas práticas:
  - Turma A1N (Prof. Diogo):
    terças 21-23h 407-2 semanal
  - Turma A2N (Prof. Emilio):
    terças 21-23h 409-2 semanal
- Plantão de dúvidas:
  - Prof. Diogo:
    terças 19-21h 528-2 semanal
  - Prof. Emilio:
    terças 17-19h 531-2 semanal
- Worksite TIDIA-AE: `PP-2018Q2-N`
  Material de aula. Entrega de atividades avaliativas.
- Repositório BitBucket: `https://bitbucket.org/diogo_martins/pp-2018q2-n/src`
  Códigos de apoio às aulas.
- Workspace Slack. Usar link de convite: `https://bit.ly/2xLp7kp`
  Discussões, dúvidas, negociação de grupos, etc.
- Atendimento:
  1. Por email: {`santana.martins, e.francesquini`}`@ufabc.edu.br`
  2. Presencialmente: no plantão de dúvidas, ou com aviso prévio via email se for fora do plantão.

2 Descrição da disciplina

Ao longo da história da computação, a combinação de diferentes domínios de aplicação, arquiteturas de computadores e metodologias de desenvolvimento de software, contribuiu com um amplo ecossistema de linguagens de programação, o qual mantém-se em constante evolução. No estudo dos conceitos de linguagens de programação, é recorrente a abordagem de reuní-las em *paradigmas*, isto é, um grupo de linguagens que compartilham características fundamentais. As linguagens de programação atualmente majoritárias são essencialmente multiparadigma, muitas englobando características de programação funcional, concorrente, ou mesmo lógica, permitindo ao programador selecionar a característica mais adequada à solução de um problema. O objetivo da disciplina é ampliar o vocabulário de programação do estudante, apresentando conceitos dos paradigmas funcional, lógico e concorrente. Ao final da disciplina, espera-se que o aluno seja capaz de analisar criticamente as características de diferentes linguagens de programação e de combinar essas características na programação multiparadigma.

3 Requisitos recomendados

A disciplina requer familiaridade com projeto de algoritmos, programação de computadores e abstração de tipos de dados. Na matriz curricular do BCC, tal conhecimento mínimo é satisfeito por quem cursou e foi aprovado em:

- Processamento da Informação
- Programação Orientada a Objetos

É desejável também a familiaridade com estruturas de dados básicas, como listas, filas, pilhas e árvores binárias, bem como noções básicas de análise de algoritmos.

4 Objetivos

- Construir um panorama dos principais paradigmas de programação;
- Compreender as características fundamentais dos paradigmas funcional, lógico e concorrente;
- Ser capaz de combinar diferentes paradigmas de programação.

5 Ementa

- Visão comparativa entre os paradigmas de programação;
- Paradigma funcional;
- Paradigma lógico;
- Programação concorrente.

6 Bibliografia

O curso fundamenta-se no estudo dos conceitos de programação declarativa, em especial a programação funcional e a programação lógica, as quais servirão de base para a análise de outras características importantes, como por exemplo a orientação a objetos e a concorrência. Com base nessa estratégia, o curso adotará como base a linguagem funcional Lisp, no dialeto Racket, o qual é multiparadigma. Para programação lógica, veremos a linguagem Prolog. Os dois livros da bibliografia principal são introdutórios e se baseiam em dialetos Lisp:

- Abelson, Harold, and Gerald Jay Sussman. Structure and interpretation of computer programs. 2nd ed. (1996). URL: `https://mitpress.mit.edu/sicp/full-text/book/book.html`. MIT Press.

- Krishnamurthi, Shriram. Programming Languages: Application and Interpretation. 2nd. ed. (2012). URL: `http://cs.brown.edu/~sk/Publications/Books/ProgLangs/`

Note que ambos os livros possuem versões online de acesso livre.

Para uma visão comparativa dos diferentes paradigmas, bem como uma visão geral dos paradigmas lógico e funcional, utilizaremos os seguintes livros:

- Sebesta, Robert W. Concepts of programming languages. 11^{th} edition (2015). Pearson.
- Van-Roy P, Haridi S. Concepts, techniques, and models of computer programming. (2004). MIT press.

Para reforçarmos alguns conceitos de programação concorrente, adotaremos como base o livro:

- Ben-Ari, M. Principles of Concurrent and Distributed Programming. 2nd. ed. (2006). Addison-Wesley Professional.

Adicionalmente à bibliografia, cada aula poderá indicar material complementar digital, em formato de texto, vídeo ou software, ou mesmo capítulos de livros físicos.

7 Criterios de avaliação

A avaliação consiste nos componentes dados pela Equação 1, onde:

$$N_F = 0.5 \cdot N_T + 0.5 \cdot N_P \tag{1}$$

- N_F é a nota final;
- N_T é a nota de teoria;
- N_P é a nota de prática.

O conceito final será obtido de acordo com a Equação 2.

$$C_F = \begin{cases} \texttt{A}, \text{ se } N_F \in [8.5, 10.0] \\ \texttt{B}, \text{ se } N_F \in [7.0, 8.5) \\ \texttt{C}, \text{ se } N_F \in [5.5, 7.0) \\ \texttt{D}, \text{ se } N_F \in [5.0, 5.5) \\ \texttt{F}, \text{ se } N_F \in [0.0, 5.0) \\ \texttt{O}, \text{ se ausência total exceder } 25\% \end{cases} \tag{2}$$

7.1 Avaliação de Teoria

A nota de teoria é formada pela Equação 3, onde:

$$N_T = 0.4 \cdot N_{p1} + 0.6 \cdot N_{p2} \tag{3}$$

- N_{p1} é a nota da prova 1;
- N_{p2} é a nota da prova 2.

Ambas as provas serão efetuadas em sala de aula, sem auxílio de computador.

7.2 Avaliação de Prática

A nota de prática é formada pela Equação 4, onde:

$$N_P = 0.6 \cdot N_{proj} + 0.4 \cdot N_{atv} \tag{4}$$

- N_{proj} é a nota do projeto;
- N_{atv} é a nota das atividades avaliativas.

7.3 Projeto

O projeto, executado em grupo de até 4 estudantes, objetiva a construção de um sistema que aplique os conhecimentos construídos ao longo do curso. Os temas serão fechados, i.e., a cada grupo será fornecida uma especificação de sistema. Cada projeto será avaliado por meio de relatório, programa de computador e apresentação. Os temas e critérios de avaliação serão divulgados posteriormente.

7.4 Atividades avaliativas

Consistem em atividades de execução individual ou em dupla, com prazo de entrega. A depender do prazo de entrega estabelecido, as atividades poderão ocorrer em aula e/ou como tarefas extra-classe. O peso de cada atividade, a ser considerado no cômputo de N_{ativ}, será variável de acordo com a complexidade da atividade.

7.5 Mecanismos de avaliação substitutivos

A prova substitutiva será aplicada ao aluno que atender às seguintes condições simultaneamente: *i)* possuir pelo menos 75% de participação; e *ii)* possuir justificativa de ausência em uma das provas. A listagem dos documentos aceitos como justificativa consta na resolução ConsEPE nº 227[1]. A nota obtida na prova substitutiva necessariamente substituirá a prova para a qual o aluno tem justificativa. Se porventura houver justificativa para ambas P1 e P2, a segunda prova substitutiva será realizada em data a ser negociada com o professor, pois não está prevista no calendário da disciplina.

7.6 Mecanismo de recuperação

A recuperação será aplicada apenas aos alunos que tiverem conceito final D ou F. Consistirá numa prova, em formato similar às aplicadas na teoria. O conteúdo da prova englobará todos os temas vistos durante o quadrimestre.

Com base na nota obtida na prova de recuperação (N_R), obtemos a nota final com recuperação (N_{FR}), que consiste na média estabelecida pela Equação 5.

$$N_{FR} = \frac{N_F + N_R}{2} \tag{5}$$

O conceito final obtido na recuperação (C_{FR}), substituirá o conceito original (C_F), de acordo com os limiares para a nota final de recuperação (N_{FR}) dados pela Equação 6.

$$C_{FR} = \begin{cases} \texttt{C}, \text{ se } N_{FR} \geq 5.5 \\ \texttt{D}, \text{ se } N_{FR} \in (5.0, 5.5) \\ \texttt{F}, \text{ se } N_{FR} \leq 5.0 \end{cases} \tag{6}$$

8 Cronograma de aulas

O plano a seguir pode variar de acordo com o aproveitamento aferido nas turmas durante o quadrimestre.

[1] `http://prograd.ufabc.edu.br/normas`

Sem.	Prática		Teoria	
	Dia	**Tema**	**Dia**	**Tema**
1	05/06	Apresentação do plano de ensino. Configuração do ambiente Racket	07/06	Introdução aos paradigmas de programação. Funcional I: fundamentos
2	12/06	Funcional I: tutorial Racket e exercícios introdutórios	14/06	Funcional II: amarrações, condicionais e procedimentos
3	19/06	Funcional II: exercícios	21/06	Funcional III: avaliação, amarração local, ambientes e fechos, repetição, recursão na cauda
4	26/06	Funcional III: exercícios	28/06	Funcional IV: estado, mutabilidade, objetos, entrada e saída
5	03/07	Funcional IV: exercícios	05/07	Funcional V: listas, funções de alta ordem, estruturas de dados funcionais
6	10/07	Funcional V: exercícios	12/07	P1
7	17/07	Revisão funcional, correção P1	19/07	Funcional VI: macros, casamento de padrões
8	24/07	Funcional VI: exercícios	26/07	Funcional VII: streams, preguiça, continuações
9	31/07	Funcional VII: exercícios	02/08	Programação lógica I: fundamentos, unificação e backtracking
10	07/08	Programação lógica I: exercícios	09/08	Programação lógica II: operadores, repetições e listas
11	14/08	Programação lógica II: exercícios	16/08	P2
12	21/08	Apresentação de projetos	23/08	Sub
13	28/08			Rec

9 Código de honra

A aprovação na disciplina é baseada no esforço e trabalho pessoal do discente, ao qual cabe garantir que não ajudará ou receberá ajuda não-permitida em qualquer atividade usada pela equipe docente para fins de avaliação (e.g. provas, trabalhos, listas, etc.).

Exemplos de violação do código de honra incluem:

- Copiar atividades avaliativas (e.g. listas, trabalhos, provas, etc.) ou permitir que outros discentes copiem suas atividades avaliativas;
- Colaboração não-permitida entre indivíduos ou grupos (e.g. oferecer vantagens em troca de soluções prontas, doar trechos para o trabalho de outro grupo, etc.);
- Permitir que outros assumam sua identidade em atividades avaliativas (e.g. entregar trabalho que não fez ou permitir que outros façam provas por você);
- Plagiarismo (i.e. aplicável a textos, programas de computador, etc.);
- Receber ou conceder ajuda em atividades avaliativas quando o contexto mostra que não é sensato receber tal ajuda.

Como consequências de violação do código de honra tem-se:

- Reprovação automática na disciplina, com conceito F;
- Denúncia na Comissão de Transgressões Disciplinares Discentes da Graduação, a qual decidirá sobre a punição adequada à violação, o que pode levar a advertência, suspensão ou desligamento, de acordo com os arts. 78-82 do Regimento Geral da UFABC.