Plano de Desenvolvimento da Disciplina

MC346 - Paradigmas de Programação Instituto de Computação Universidade Estadual de Campinas		2º semestre de 2025 Turma A Professor: André Santanchè

Horários

Segunda: 16:00 às 18:00

Quarta: 16:00 às 18:00

Atendimento

O professor, o PED e o PAD estarão disponíveis para atendimento. O atendimento deverá ser agendado, conforme orientado no ambiente da disciplina.

Ementa

Visão comparativa de paradigmas de programação: programação funcional, programação lógica e programação rápida (prototipação).

Programa

Página da Disciplina

https://www.ic.unicamp.br/~santanche/teaching/paradigms/2025-2/

Eixos de Estudo de Linguagens e Paradigmas

Este curso está organizado em dois eixos de análise.

O primeiro analisa dimensões de abstração e estrutura das linguagens, envolvendo aspectos como: abstrações com dados e processos, modularidade e manutenção de estado e primitivas de primeira ordem. Este eixo segue a linha de (Abelson et al., 1996).

O segundo eixo analisa como as linguagens e respectivos paradigmas adotam abstrações e organizam estruturas. Para cada paradigma, serão escolhidas linguagens representativas a partir de indicadores como o TIOBE (https://www.tiobe.com/tiobe-index/) e estudos como Top Programming Languages 2024 (https://spectrum.ieee.org/top-programming-languages-2024).

Linguagens estudadas: C++, Java, Python, Clojure, Prolog, JavaScript e React.

Programa do Curso

Os tópicos a seguir não estão organizados em ordem cronológica, já que o primeiro e o segundo eixos são trabalhados concomitantemente.

Fundamentos

- O que é uma linguagem de programação.
- O que é um paradigma.

Primeiro Eixo - Primitivas e Abstrações da Linguagem

- Abstrações matemáticas da computação:
  - Noções de linguagens formais e a máquina de Turing.
    - Linguagens Turing completas.
  - Cálculo lambda.
  - Tipo abstrato de dados.
- Metalinguística e Meta-programação.
- Primitivas de primeira ordem: funções, classes, objetos e regras.
- Estado, escopo e closure.
- Lazy versus Eager (Strict) evaluation.
- Linguagens minimalistas (RISC-like) e maximalistas (CISC-like).

Segundo Eixo - Paradigmas e Linguagens

Imperativo/Procedural

Visão geral dos fundamentos na forma de revisão.

Orientado a Objetos

Visão geral dos fundamentos na forma de revisão. A orientação a objetos sob duas perspectivas: forma de abstrair o universo de discurso; abordagem de modularização.

Diferentes abordagens dentro do paradigma:
- Tipagem forte e fraca.
- Relações entre objetos e classes.
- Programação Rápida e Prototipação.
- Exemplos em: Java, Python e JavaScript.

Declarativas

Funcional

Fundamentos de linguagens funcionais e contraste com procedurais: "o que" resolver versus "como" resolver.

- Funções como primitivas de primeira ordem.
  - Relações com o cálculo lambda.
  - Funções de ordem superior.
- Imutabilidade dos dados.
- Recursividade versus loop.
- Linguagem Scheme:
  - Abordagem minimalista do Scheme versus maximalista do JavaScript.
  - Construindo uma linguagem (meta-programação).
- Funções versus Classes em JavaScript.

Lógica

Fundamentos de linguagens declarativas e contraste com procedurais e funcionais: espectro "o que" resolver versus "como" resolver.

- Diferentes abordagens declarativas:
  - Queries e regras.
- Computação centrada no conhecimento.
  - Conhecimento e Prolog.

- Fundamentos de Grafos de Conhecimento e Ontologias.
- Solução de problemas por inferência.
- Inferências na biologia.

Concorrente e Paralelo

Introdução ao princípio de concorrência e paralelismo e seu impacto no clássico fluxo de controle.

- Modelos:
  - Abordagem de threads/memória compartilhada.
    - Clojure e dados persistentes e imutáveis.
  - Abordagem de passagem de mensagens/atores.
    - Passagem de mensagens em Clojure.

Reativo e Orientado a Eventos

Programação orientada a eventos e fluxos reativos.

- Programação assíncrona e não bloqueante.
- Loop de eventos.
- Promessas e callbacks.

Critérios de Avaliação

A disciplina será pautada em pequenos projetos práticos desenvolvidos em sala trabalhos executados em equipe, envolvendo duas linhas:

- A primeira será mais voltada ao primeiro eixo, explorando aspectos que envolvem os paradigmas
- A segunda será voltada ao segundo eixo e linguagens de mercado e relevantes dentro dos paradigmas.

O curso terá as seguintes avaliações e respectivos períodos:

Legenda	Descrição	Quando ocorre
Pa	Participação no curso	semanalmente, até 19/11/2025
Lab	Trabalhos de Laboratório	semanalmente, até 19/11/2025
Prj	Projetos em Equipe	entregas parciais a serem combinadas durante o semestre; apresentação final até 19/11/2025

A participação no curso (Pa) está associada a tarefas individuais de participação que serão entregues ao final das aulas.

A especificação dos Projetos em Equipe (Prj) serão entregues em documento específico. Esse trabalho terá datas de entrega parciais que serão definidas no ambiente virtual durante o curso.

Trabalhos de laboratório (Lab) desenvolvidos durante o curso e sempre são lançados no horário de laboratório. Eventualmente, atividades de laboratório podem ser vinculadas ao projeto final. Nesses casos, a distribuição das notas (entre Lab e Prj) será especificado com o trabalho.

A média é calculada como segue:

$$\text{média} = (Pa + Lab * 4 + Prj * 5) / 10$$

Exame final

- Estarão dispensados do exame apenas os alunos com $média_{se}$ >= 5
- Data de realização: 10/12/2023
- Para estar habilitado a realizar o exame o aluno deve ter média mínima: $média_{se}$ >= 2,5
- Neste caso o cálculo da média para alunos que precisam do exame:

$$média_{final} = (média_{se} + nota_{exame}) / 2$$

Bibliografia

Abelson, H., Sussman, G. J., & Sussman, J. (1996). **Structure and Interpretation of Computer Programs** (2nd ed.). MIT Press. https://mitp-content-server.mit.edu/books/content/sectbyfn/books_pres_0/6515/sicp.zip/index.html

Dybvig, R. K. (2009). **The Scheme Programming Language** (4th ed.). MIT Press. https://www.scheme.com/tspl4/

Kiczales, G., des Rivieres, J., & Bobrow, D. G. (1991). The Art of the Metaobject Protocol. In The Art of the Metaobject Protocol. The MIT Press. https://doi.org/10.7551/MITPRESS/1405.001.0001

Meyer, Bertrand (2000). **Object-Oriented Software Construction**, 2a edição. Prentice Hall.