

Campus: São José dos Campos		
Curso (s): Bacharelado em Ciência da Computação/ Engenharia da Computação/ Matemática Computacional		
Unidade Curricular (UC): Compiladores		
Unidade Curricular (UC): <i>[nome da UC em inglês]</i> Compilers		
Código da UC: 2615		
Docente responsável: Luis A. M. Pereira		Contato (e-mail): <i>[opcional]</i>
Ano letivo: 2022	Termo: sexto	Turno/Turma: Integral
Nome do Grupo/Módulo/Eixo da UC (se houver):		Idioma predominante em que a UC será oferecida: <input checked="" type="checkbox"/> Português <input type="checkbox"/> English <input type="checkbox"/> Español <input type="checkbox"/> Français <input type="checkbox"/> Libras <input type="checkbox"/> Outro:
UC: <input checked="" type="checkbox"/> Fixa <input type="checkbox"/> Eletiva <input type="checkbox"/> Optativa	Oferecida como: <input checked="" type="checkbox"/> Disciplina <input type="checkbox"/> Módulo <input type="checkbox"/> Estágio <input type="checkbox"/> Outro:	Oferta da UC: <input checked="" type="checkbox"/> Semestral <input type="checkbox"/> Anual
Ambiente Virtual de Aprendizagem: <input type="checkbox"/> Moodle <input checked="" type="checkbox"/> Classroom <input type="checkbox"/> Outro: <input type="checkbox"/> Não se aplica		
Pré-Requisito (s) - Indicar Código e Nome (s) da (s) UC: 2616 - Linguagens Formais e Autômatos		
Carga horária total (em horas): 72		
Carga horária teórica (em horas): 36	Carga horária prática (em horas): 36	Carga horária de extensão (em horas, se houver):
Se houver atividades de extensão, indicar código e nome do projeto ou programa vinculado na Pró-Reitoria de Extensão e Cultura (ProEC):		
Ementa: <i>Sistema de Varredura - Análise Léxica; Gerador de Analisador Léxico; Análise Sintática Descendente; Análise Sintática Ascendente; Gerador de Analisador Sintático; Análise Semântica; Geração de Código; Otimização de Código.</i>		
Conteúdo programático: <i>Introdução: Importância dos compiladores; histórico e evolução; visão geral do processo de compilação. Partida rápida e transposição. Análise Léxica: Uso de expressões regulares para descrição de padrões de tokens; algoritmo de Thompson; construção de subconjuntos; otimização dos autômatos finitos determinísticos; transformação de autômatos finitos em programas de reconhecimento de cadeias; gerador de analisadores léxicos (Flex). Análise Sintática: Árvores sintáticas; análise sintática descendente; análise sintática ascendente; gerador de analisadores sintáticos (YACC-Bison). Análise Semântica: Algoritmos para computação de atributos; tabela de símbolos; tipos de dados e verificação de tipos. Geração de Código: Código intermediário (código de três endereços); geração de código para referências e estruturas de dados; geração de código para declarações de controle e expressões lógicas; geração de código para chamadas de procedimentos e funções. Otimização de Código: Escolha de modos de endereçamento. Substituição de instruções. Eliminação de operações redundantes.</i>		
Objetivos: <u>Gerais:</u> A disciplina tem o objetivo de apresentar os conceitos fundamentais sobre compiladores, por meio de abordagem teórica e prática. <u>Específicos:</u> Apresentar aos alunos técnicas consolidadas de projeto e construção de compiladores; Capacitar os alunos para a especificação e utilização de gramáticas usadas na construção de compiladores; Habilitar os alunos a compreender as fases de análise léxica, sintática e semântica; Capacitar os alunos para o uso de geradores automáticos de analisadores léxicos e sintáticos; Apresentar aos alunos uma visão geral do processo de síntese realizado por um compilador; Proporcionar aos alunos a experiência de projetar e construir um compilador.		

Metodologia de ensino:

Aulas expositivas sobre a fundamentação teórica dos assuntos; Exercícios em classe (ou laboratório) e extra-classe sobre os assuntos abordados em aula; Videoaulas; Desenvolvimento de um compilador.

Avaliação: Serão adotados os seguintes instrumentos de avaliação:

- 1 prova escrita (peso 4).

- 1 projeto da disciplina – trabalho final (peso 6): o projeto será desenvolvido individualmente ou em duplas, dependendo do tamanho da turma. A avaliação do projeto será feita mediante a apresentação do projeto para o professor. O projeto consistirá no desenvolvimento das fases de análise de um compilador.

A nota final será calculada da seguinte forma: $NF = (4 * Prova + 6 * Projeto) / 10$

A promoção do aluno na UC obedecerá aos critérios estabelecidos pela Pró-Reitoria de Graduação, tal como discutido no projeto pedagógico do curso.

Bibliografia:

Básica:

1. Louden, Kenneth C; Silva, Flávio S.C. Compiladores: princípios e práticas. São Paulo: Thomson, 2004. 569 p. ISBN 978-85-221-0422-2.
2. Aho, Alfred V et al. Compiladores: princípios, técnicas e ferramentas. 2a ed. São Paulo: Person Addison Wesley, 2007. 634 p. ISBN 978-85-88639-24-9. tradução de "Compilers: principles, techniques, and tools.
3. Appel, Andrew W; Palsberg, Jens. Modern compiler implementation in Java. 2 ed. New York: Cambridge at the University Press, 2002. 501 p. ISBN 978-0-521-82060-8.
4. Ricarte, I. Introdução à Compilação. Editora Elsevier/Campus, 2008.

Complementar:

1. SCOTT, Michael L. Programming language pragmatics. New York: Morgan Kaufmann, c2009. 910 p. ISBN 978-0-12-374514-9.
2. HOPCROFT, John E; MOTWANI, Rajeev; ULLMAN, Jeffrey D. Introdução à teoria de autômatos, linguagens e computação. [Introduction to automata theory, languages, and computation.]. Rio de Janeiro: Campus, 2002. 560 p. ISBN 978- 85-352-1072-9.
3. ULLMAN, Jeffrey D; MOTWANI, Rajeev; HOPCROFT, John E. Introduction to automata theory, languages, and computation. 3.ed. Boston (USA): Pearson, 2006. 535 p. ISBN 978-0-321-45536-9.
4. Ricarte, Ivan. Introdução à compilação. Rio de Janeiro: Elsevier, 2008. 264 p. ISBN 9788535230673.
5. PRICE, Ana Maria de Alencar; TOSCANI, Simão Sirineo. Implementação de linguagens de programação: compiladores. 3.ed. Porto Alegre: Bookman, 2008. 9. 195 p. ISBN 978-85-7780-348-4.