PLANO DE ENSINO

1. IDENTIFICAÇÃO

Curso: Ciência da Computação
Componente Curricular: Construção de Compiladores
Fase: 6 (seis)
Ano/Semestre: 2012/2
Numero de Créditos: 4 (quatro)
Carga horária - Hora Aula: 72 horas
Carga horária - Hora Relógio: 60 horas
Professor: Braulio Adriano de Mello

2.Objetivo Geral do Curso

O curso tem por objetivo a formação integral de novos cientistas e profissionais da computação, os quais deverão possuir conhecimentos técnicos e científicos e serem capazes de aplicar estes conhecimentos, de forma inovadora e transformadora, nas diferentes áreas de conhecimento da Computação. Adicionalmente, os egressos do curso deverão ser capazes de adaptar-se às constantes mudanças tecnológicas e sociais, e ter uma formação ao mesmo tempo cidadã, interdisciplinar e profissional

3. EMENTA

Projeto de especificação de linguagens de programação. Etapas que compreendem o processo de compilação: Análise Léxica, Análise Sintática, Análise Semântica, Geração e Otimização de Código. Evolução e tendências da área de compiladores e linguagens de programação. Implementação de analisadores.

4. JUSTIFICATIVA

A disciplina de Construção de Compiladores permite ao estudante compreender conceitos e características do processo de compilação. Oferece melhores condições para avaliar as propriedades de uma linguagem e seus efeitos nos códigos gerados considerando, por exemplo, a relação com chamadas ao sistema (sistemas operativos), desdobramentos dos conceitos de linguagens de programação e computabilidade. Podem ser também enumeradas benefícios indiretos resultantes de conhecimentos em torno do processo de compilação. Entre eles, a disponibilidade de referenciais ou informações adicionais que interferem positivamente nos métodos ou práticas de programação.

5. OBJETIVOS

5.1. GERAL:

Compreender a estrutura de compiladores, o processo de compilação no reconhecimento de linguagens e a geração/otimização de código, construção de analisadores léxicos e sintáticos.

5.2. ESPECÍFICOS:

- Estudo da Estrutura e Processo de compilação
- Entender e construir analisadores observando formalismos e algoritmos utilizados para os diferentes tipos de máquinas reconhecedoras
- Estudo e experimentação de soluções para verificação semântica e otimização de código
- Desenvolver um projeto prático atendendo etapas do processo de compilação incentivando os estudantes a realizar esforços integradores com demais áreas do curso, preferencialmente observando os CCRs em curso no semestre;
- Fortalecer iniciativas tais como elaboração de novos materiais de apoio e dinâmicas alternativas de atividades buscando melhorar as condições didático-pedagógicas para aprendizado de compiladores.

6. CRONOGRAMA E CONTEÚDO PROGRAMÁTICO

Horas	Número Encontro	Conteúdo
3	1	Apresentação da disciplina, apreciação do plano de ensino e sistema de avaliação
2	2	Revisão Linguagens Formais e Autômatos
3	3	Componentes de um Compilador e processo de compilação: Analisadores Léxico, Sintático e Semântico e o Gerador de Código.
2	4	Componentes de um Compilador e processo de compilação: Analisadores Léxico, Sintático e Semântico e o Gerador de Código.
3	5	Interpretação/compilação, analisadores, geração e otimização de código
2	6	Interpretação/compilação, analisadores, geração e otimização de código
3	7	Construção de analisadores léxicos: autômato finito como máquina reconhecedora
2	8	Estudo de caso: Lex e yacc
3	9	**Trabalho 1:** Projeto e implementação de analisador léxico
2	10	Projeto e implementação de analisador léxico
3	11	Análise sintática: Analisadores ascendentes e descendentes, simplificação de GLCs, árvores de derivação (esq., dir.), analisadores LR e LL
2	12	Análise sintática: Analisadores ascendentes e descendentes, simplificação de GLCs, árvores de derivação (esq., dir.), analisadores LR e LL
3	13	Autômatos de Pilha como máquina reconhecedora.
2	14	Autômatos de Pilha como máquina reconhecedora.
3	15	**Apresentação do trabalho 1**
2	16	**Avaliação 1**
3	17	Análise sintática: precedência de operadores (shift-reduce)
2	18	Implementação de analisador sintático da gramática de operadores com shift-reduce (**Trabalho facultativo**).
3	19	Análise sintática SLR: construção de itens válidos e conjunto de transições (gramática de operadores)
2	20	Análise sintática SLR: construção da tabela de parsing
3	21	Algoritmo de reconhecimento sintático SLR.
2	22	**Trabalho 2:** Implementação de reconhecedor sintático SLR

3	23	Implementação de reconhecedor sintático SLR
2	24	Análise Semântica e Código intermediário: conceitos, características, tradução dirigida por sintaxe.
3	25	Otimização de código: conceitos, características, otimização por Grafos Acíclicos Dirigidos.
2	26	Implementação de otimizador de código (**trabalho facultativo**)
3	27	Código objeto e editor de ligação
2	28	**Avaliação 2**
3	29	**Apresentação do trabalho 2**
2	30	**Prova substitutiva –** recuperação de rendimento (todo o conteúdo visto na disciplina).
75		

Obs.: cronograma e/ou conteúdos podem sofrer modificações durante o semestre

7. PROCEDIMENTOS METODOLÓGICOS (estratégias de ensino, equipamentos, entre outros)

Embora a disciplina contemple conteúdo a ser trabalhado em encontros teóricos, maior esforço é dedicado ao processo de construção de compiladores. Deste modo, serão contempladas aulas expositivas/dialogadas para atendimento de conceitos específicos, tais como para máquinas reconhecedoras e otimização, seguidos de maior dedicação às atividades práticas no projeto e implementação dos processos que atendem a estrutura de compiladores. O perfil aplicado da disciplina aponta para maior ênfase em exercícios práticos, demonstrações, contextualização experimentação e implementação.

8. AVALIAÇÃO DO PROCESSO ENSINO-APRENDIZAGEM

Instrumentos de avaliação: provas teóricas, avaliação escrita em aula, exercícios extra-classe, trabalhos de implementação, elaboração de texto/artigo, seminários entre outros trabalhos de complexidade variada.

Avaliação de trabalhos práticos: 50% da nota pela parte escrita (grupo) e 50% da nota de acordo com avaliação individual (independente do grupo de trabalho).

Trabalho Facultativo (TF): Utilizado apenas para recuperação de nota, quando necessário. O acréscimo de nota é limitado a 5 pontos, num total de 100, proporcionalmente ao número de TFs concluídos. Novos TFs podem ser definidos durante o semestre. Trata-se de trabalho individual. Pode ser entregue até o último dia letivo de aula.

Avaliações da disciplina:

NP1: Avaliação 1 (0,5) e Trabalho 1 (0,5)

NP2: Avaliação 2 (0,5) e Trabalho 2 (0,5)

Prova de recuperação de rendimento: substitui a menor nota dentre as avaliações, notas dos trabalhos não são passíveis de substituição

Horário de atendimento dos estudantes: Segunda-feira; 10:10 as 11:30

9. REFERÊNCIAS

9.1. BÁSICAS:

PRICE, A. M. A., TOSCANI, S. S. Implementação de Linguagens de Programação: Compiladores. Bookman Companhia Ed., 2008.

HOPCROFT, J. F., ULLMAN, J. D., Motwani, R., “Introdução a teoria dos

automatos", Ed. Campus, 2002.
AHO, A. V., SETHI, R.,LAM,M., "Compiladores: Princípios, técnicas e ferramentas", Ed. Longman do Brasil, 2007.
GRUNE, D., BAL, H. E., JACOBS, C., LANGENDOEN, K. Projeto Moderno De Compiladores: implementação e aplicações. Rio de Janeiro: Campus, 2001.

9.2. COMPLEMENTARES:

WOOD, D. , **"Theory of Computation"**, Ed. John Wiley & Sons, 1987.
FURTADO, O. J. V., **"Apostila de Linguagens Formais e Compiladores – versão 2"**, UFSC, 2002.
DELAMARO, M. E. **Como Construir Um Compilador Utilizando Ferramentas**
Java. Rio de Janeiro: NOVATEC, 2004.
LOUDEN, K. C. **Compiladores Princípios e Práticas**. São Paulo: THOMSON PIONEIRA, 2004.
Furtado, Olinto. Linguagens formais e compiladores. <www.inf.ufsc.br/~olinto/apostila-lfc.doc>. Acesso em 30/09/2012.