UNIVERSIDADE FEDERAL DA FRONTEIRA SUL

PLANO DE ENSINO

1. IDENTIFICAÇÃO

Curso: Ciência da Computação - Diurno
Componente Curricular: Construção de Compiladores
Fase: 7 (sete)
Ano/Semestre: 2015/1
Numero de Créditos: 4 (quatro)
Carga horária - Hora Aula: 72 horas
Carga horária - Hora Relógio: 60 horas
Professor: José Carlos Toniazzo
Atendimento ao Aluno: Quintas entre 10:10 e 12:10

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. OBJETIVOS

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.

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.

5. CRONOGRAMA E CONTEÚDOS PROGRAMÁTICOS

Encontro	Conteúdo
1	Apresentação da disciplina, apreciação do plano de ensino e sistema de avaliação
2	Linguagens Formais e Autômatos
3	Componentes de um Compilador e processo de compilação: Analisadores Léxico, Sintático e Semântico e o Gerador de Código.
4	Componentes de um Compilador e processo de compilação: Analisadores Léxico, Sintático e Semântico e o Gerador de Código.
5	Interpretação/compilação, analisadores, geração e otimização de código
6	Interpretação/compilação, analisadores, geração e otimização de código
7	Construção de analisadores léxicos: autômato finito como máquina reconhecedora
8	Estudo de caso: Lex e yacc
9	**Trabalho 1:** Projeto e implementação de analisador léxico
10	Projeto e implementação de analisador léxico
11	Análise sintática: Analisadores ascendentes e descendentes, simplificação de GLCs, árvores de derivação (esq., dir.), analisadores LR e LL
12	Análise sintática: Analisadores ascendentes e descendentes, simplificação de GLCs, árvores de derivação (esq., dir.), analisadores LR e LL
13	Autômatos de Pilha como máquina reconhecedora.
14	Autômatos de Pilha como máquina reconhecedora.
15	**Apresentação do trabalho 1**
16	**Avaliação 1**
17	Análise sintática: precedência de operadores (shift-reduce)
18	Implementação de analisador sintático da gramática de operadores com shift-reduce (**Trabalho facultativo**).
19	Análise sintática SLR: construção de itens válidos e conjunto de transições (gramática de operadores)
20	Análise sintática SLR: construção da tabela de parsing
21	Algoritmo de reconhecimento sintático SLR.
22	**Trabalho 2:** Implementação de reconhecedor sintático SLR
23	Implementação de reconhecedor sintático SLR
24	Análise Semântica e Código intermediário: conceitos, características, tradução dirigida por sintaxe.
25	Análise Semântica e Código intermediário: conceitos, características, tradução dirigida por sintaxe.
26	Otimização de código: conceitos, características, otimização por Grafos Acíclicos Dirigidos.
27	Implementação de otimizador de código (**trabalho facultativo**)
28	Código objeto e editor de ligação
29	**Avaliação 2**
30	**Apresentação do trabalho 2**
31	**Prova substitutiva –** recuperação de rendimento (todo o conteúdo visto na disciplina).

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

6. PROCEDIMENTOS METODOLÓGICOS

Conduzir a disciplina com aulas expositivas/dialogadas enquanto discutidos os itens de cunho teórico, evoluindo em tópicos específicos para exercícios práticos, demonstrações, contextualização. Atividades práticas em implementação e experimentação.

7. AVALIAÇÃO DO PROCESSO ENSINO-APRENDIZAGEM

Uso de abordagens tais como: provas teóricas, avaliação escrita em aula, exercícios extra-classe, trabalhos de implementação, elaboração de texto/artigo, seminários entre outros trabalhos com complexidade variada.
Avaliação de trabalhos: 50% da nota pela parte escrita (grupo) e 50% da nota em avaliação individual (apresentação).

Avaliações da disciplina:
NP1: uma prova teórica (0,6) e um trabalho de implementação (0,4)
NP2: uma prova teórica (0,6) e um trabalho de implementação (0,4)

7.1 RECUPERAÇÃO: NOVAS OPORTUNIDADES DE APRENDIZAGEM E AVALIAÇÃO

Ao final do semestre, caso não tenha sido alcançada pontuação suficiente para aprovação, o estudante poderá usufruir de duas oportunidades de recuperação:
a) Avaliação escrita adicional (todo o conteúdo do semestre) cuja nota será utilizada para substituir a menor nota dentre as avaliações regulares.
b) Trabalhos facultativos individuais que, se entregues até o último dia letivo de aula da disciplina, podem ser utilizados para aumento da média final limitado a 30% da nota original. Trabalhos não realizados não implicam em redução de nota.

8. REFERÊNCIAS

8.1 BÁSICA

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.

8.2 COMPLEMENTAR

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.

8.3 SUGESTÕES

_____ Professor

_____ Coordenador do curso