Universidade Federal de Santa Catarina
Departamento de Engenharia Elétrica
Plano de Ensino 2017-2

Disciplina: Computação Científica I – EEL 7021
Carga horária: 72 horas aula (02 horas aula/semana = teórica)
(02 horas aula/semana = prática)
Período: 2017-2
Turmas: 02202, 02235
Professor: Samir Ahmad Mussa (Teórica e Laboratório)

Informações Diversas

E-mail de contato do professor: samir@inep.ufsc.br.
Horário de atendimento pelo professor:
Prof. Samir Ahmad Mussa: Matutino e vespertino no INEP sala 213, fone: 3721-7464. Gentileza consultar e agendar o horário de atendimento com antecedência.

Horário e local de atendimento do monitor: a ser definido na página da disciplina.
Horário e local das Aulas:
Teoria_____3.1010-2 (02202, 02235)__ sala CTC305
Laboratório:__3.1620-2 (2202A)_____ sala LIICT8
Laboratório:__2.0910-2 (02235)_____ sala LIICT6

Ementa

Princípios gerais de informática; princípios gerais de concepção de programas; técnicas de modularização; programação orientada a objeto; linguagens de programação; aplicação de uma linguagem de alto nível; paralelização de algorítmos; noções de processamento distribuído.

Objetivos

Introdução à utilização de computadores como ferramenta de desenvolvimento de programas aplicados a problemas de engenharia e ciência. Esta disciplina é direcionada aos alunos de cursos de engenharia, e visa proporcionar ao estudante um primeiro contato com programação utilizando linguagem de alto nível. Deseja-se que os estudantes desenvolvam as seguintes habilidades: (1) capacidade de analisar um problema de engenharia, (2) capacidade de propor soluções na forma de um algoritmo e (3) capacidade de implementá-lo na forma de uma linguagem específica de programação de alto nível. Serão enfatizados os conceitos associados à programação de computadores em geral, assim como às particularidades da linguagem de programação de alto nível escolhida.

Conteúdo Programático e Carga Horária

1. Princípios gerais de informática, arquitetura e algoritmos (2 horas);
2. Ambiente de programação (2 horas)
3. Introdução à programação em C (4 horas);
4. Programação estruturada (4 horas);
5. Técnicas de controle de programa (4 horas);
6. Funções (4 horas);
7. Manipulação de matrizes (4 horas);
8. Ponteiros (8 horas);
9. Manipulação de caracteres e *strings* (4 horas);
10. Estruturas uniões, manipulação de bits e enumerações (4 horas);

11. Manipulação de arquivos (4 horas);
12. Estruturas de dados (4 horas);
13. Introdução à programação orientada a objetos (2 horas);
14. Noções de processamento distribuído e paralelização de algoritmos (2 horas);
15. Aplicações científicas e exercícios (12 horas).
16. Atividades de revisão e avaliações (8 horas).

Sistema de Avaliação

O desempenho do estudante será avaliado através de duas provas teóricas e de uma avaliação prática em laboratório. Também fazem parte da avaliação as atividades em laboratório e exercícios aplicados em sala de aula. A nota final será composta pela seguinte ponderação:

Avaliação teórica I	25%
Avaliação teórica II	45%
Avaliação Prática I	10%
Avaliação Prática II	10%
Atividades em laboratório/exercícios/Trabalhos	10%

Se, ao final da disciplina, o aluno não atingir a nota mínima de 6,0, mas possuir média igual ou superior a 3,0 e freqüência maior ou igual a 75 %, o mesmo poderá realizar uma prova de recuperação referente a todo o conteúdo da disciplina. A nota final será a média entre a nota obtida ao longo do semestre e a nota da recuperação.

Datas Importantes

Avaliação teórica I	26/SET
Avaliação teórica II	14/NOV
Avaliação Prática I	SETEMBRO
Avaliação Prática I	NOVEMBRO

Livro Texto

C Completo e Total, H. Schildt, Makron Books, terceira edição, 1997.

Página da Disciplina:

http://moodle.ufsc.br

Bibliografia

1. Como Programar em C, H.M. Deitel e P.J. Deitel, LTC Editora, segunda edição, 1999.
2. C the Complete Reference, H. Schildt, 3rd edition, 1997/2006.
3. Orientação a Objetos em C++, F. Montenegro e R. Pacheco, Ciência Moderna.
4. Projeto de Algoritmos com Implementações em Pascal e C, N. Ziviani, 2004.
5. Algoritmos Estruturados, H. Farrer e outros, Editora LTC, 1999.

Programa da Disciplina:

Princípios gerais de informática, arquitetura e algoritmos:
História dos computadores; Arquitetura básica: CPU, memória, armazenagem, dispositivos de entrada e saída, periféricos; Instruções, programas e algoritmos; Processo de edição, compilação, linkagem e execução.

Elementos de uma linguagem de alto nível: variáveis, estruturas e funções:
Organização de um programa em ANSI-C; Declaração de variáveis, tipos de dados; Instruções e chamadas de funções da biblioteca padrão.

Atribuição, operações com variáveis, retorno de funções e passagem de parâmetros:
Atribuição e operações com variáveis e constantes; tipos de funções e retorno de valor; passagem por valor e por referência.

Estruturas de controle de fluxo: seleção e repetição:
Condições e estruturas de controle; Fluxogramas de estruturas; Seleção de blocos usando ``if'' e ``switch''; Repetição condicional de blocos usando: ``for'', ``while'' e ``do-while''; Encadeamento de blocos.

Introdução a ponteiros, alocação de memória e estruturas de dados:
Ponteiros, endereços, dados e representação na memória; Tipos de alocação de memória: estática e dinâmica; Áreas de memória: ``stack'', ``heap'' e instruções; Gerenciamento e monitoramento de memória alocada; Funções de alocação e liberação de memória ANSI C: malloc(), free(); Alocação estática de vetores e matrizes; Alocação dinâmica de vetores e matrizes; Passagem e retorno de vetores e matrizes a funções.

Princípios gerais de concepção de programas e técnicas de modularização:
Organização de programa fonte: nome de variáveis e funções, formatação de blocos com tabulação; análise, desenvolvimento e testes de programa em blocos; Identificação de similaridades entre blocos de instruções; desenvolvimento com técnicas de aprimoramento sucessivo (``top-down'').

Aplicações científicas de uma linguagem de alto nível:
Experimentos em laboratório com linguagem de programação ANSI-C; Implementação de programas envolvendo: operações matemáticas básicas, cálculo de médias, operações com vetores, operações com números complexos, integração de funções, cálculo de raízes e operações matriciais.

Introdução a programação orientada a objetos (OOP):
Encapsulamento de dados utilizando struct; Exemplos de programa utilizando classes; Similaridades entre struct e classes; Conceitos básicos de OOP; Exemplos em linguagem ANSI-C++.

Noções de complexidade computacional, processamento distribuído e de paralelização de algoritmos: Exemplos de análise de complexidade de algoritmos; Exemplos de sistemas de processamento paralelo e processamento distribuído;