MINISTÉRIO DA EDUCAÇÃO
UNIVERSIDADE FEDERAL DE SANTA CATARINA
CENTRO TECNOLÓGICO
DEPARTAMENTO DE INFORMÁTICA E ESTATÍSTICA

PLANO DE ENSINO

1. Identificação

Disciplina: INE5408 - Estruturas de Dados

Nível: Graduação

Carga Horária: 108 horas-aula (Teórica: 54; Prática: 54)

Período Letivo: 2026-1

Turmas:

03208A - CIÊNCIAS DA COMPUTAÇÃO (208)
Horário: 3 15:10 3 (CTC202), 5 15:10 3 (INE101)
Docentes:
Aldo Von Wangenheim (aldo.vw@ufsc.br)
Alexandre Goncalves Silva (alexandre.goncalves.silva@ufsc.br)

03208B - CIÊNCIAS DA COMPUTAÇÃO (208)
Horário: 3 15:10 3 (CTC202), 4 15:10 3 (CTC106)
Docente: Alexandre Goncalves Silva
(alexandre.goncalves.silva@ufsc.br)

2. Ementa

Alocação dinâmica de memória. Variáveis estáticas e dinâmicas. Estruturas lineares. Tabelas de Espalhamento. Árvores. Árvores de Pesquisa. Métodos de ordenação. Métodos de acesso a arquivos. Técnicas de implementações iterativas e recursivas de estruturas de dados. Complexidade dos algoritmos em estruturas de dados.

3. Objetivos

3.1 Objetivo Geral:

Capacitar o estudante a compreender, tanto do ponto de vista conceitual e teórico quanto prático (implementação e solução de problemas reais), as estruturas de dados clássicas a partir da perspectiva de diferentes paradigmas de programação.

3.2 Objetivos Específicos:

a) Identificar o papel das estruturas de dados no desenvolvimento de software.

b) Compreender a teoria associada a cada modelo, sendo capaz de compreender e aplicar adequadamente aspectos de complexidade de algoritmos associados.

c) Capacitar o aluno a criar uma biblioteca de estruturas de dados reutilizáveis.

d) Identificar as estruturas de dados pertinentes a um problema dado.

e) Transferir seus conhecimentos a problemas práticos do mundo real, sendo capaz de solucionar estes problemas práticos do mundo através da utilização das Estruturas de Dados adequadas, tanto do ponto de vista da eficácia quanto da eficiência.

4. Conteúdo Programático

1 Alocação dinâmica de memória [6 horas-aula]
- 1.1 Variáveis estáticas e dinâmicas
- 1.2 Ponteiros
- 1.3 Passagem de parâmetros por referência, valor e nome

2 Estruturas lineares [24 horas-aula]
- 2.1 Listas
- 2.2 Pilhas
- 2.3 Filas

3 Complexidade de algoritmos [6 horas-aula]
- 3.1 Conceitos de Complexidade de Algoritmos
- 3.2 Métodos práticos para Análise da Complexidade de Estruturas e Algoritmos

4 Árvores [24 horas-aula]
- 4.1 Árvore binária
- 4.2 Árvores binárias semibalancedas (Árvore AVL, Árvore Red-Black)
- 4.3 Árvore semibalanceadas multivias (Árvore B e B+)
- 4.4 Árvores binárias multichaves (Árvore k-d)
- 4.5 Árvores semibalanceadas multivias multichaves (Árvores k-b-d e b-k-d)

5 Tabela de espalhamento (hash) [15 horas-aula]
- 5.1 Tratamento de colisões
- 5.2 Funções de espalhamento

6 Métodos de ordenação [12 horas-aula]
- 6.1 Conceitos básicos, implicações e premissas
- 6.2 Métodos de Complexidade Quadrática
  - 6.2.1 Método por inserção
  - 6.2.2 Método por seleção
  - 6.2.3 Método da bolha
- 6.3 Métodos Avançados e de Complexidade n log n
  - 6.3.1 Método Quicksort
  - 6.3.2 Método Heapsort

7 Estruturas de dados em arquivo [21 horas-aula]
- 7.1 Acesso direto e Acesso sequencial
- 7.2 Organização de Arquvos de Índices
  - 7.2.1 Métodos indexados seqüenciais, ISAM e VSAM
  - 7.2.2 Indexação por Árvore
  - 7.2.3 Indexação por Multilistas e Lista Invertida

5. Metodologia

A disciplina terá um enfoque eminentemente prático. A metodologia de ensino será baseada no contraponto entre Aulas Teóricas e Aulas Práticas. Para tanto, todo novo assunto será introduzido em uma aula teórica. As atividades práticas podem ser executadas com auxílio de um Estagiário de Docência ou Monitor, e se caracterizam por modelagem e implementação com o objetivo de fixar o conteúdo, além da discussão em grupo de problemas de compreensão e implementação encontrados pela turma. Como linguagem de programação para o ensino do conteúdo fica determinada a Linguagem C++ (compilador g++ da *GNU Compiler Collection*), sendo o ambiente VPL do Moodle utilizado como ferramenta de desenvolvimento preferencial.

Haverá uma breve introdução dos recursos dessas plataformas à medida que se tornarem necessárias, principalmente as de teste unitário. É de responsabilidade do aluno a proficiência nas mesmas. O ensino dos conteúdos se dará por meio da utilização progressiva do Paradigma Orientado a Objetos, de acordo com a adequação à melhor compreensão do conteúdo. O sistema EAD Moodle, disponível em moodle.ufsc.br, será utilizado para guiar e organizar o ensino, sendo o repositório oficial de material de aula, devendo os alunos constantemente consultá-lo para saber de novos exercícios e trabalhos requeridos. A disciplina no Moodle também detalhará o cronograma deste plano de ensino, servindo para marcar os prazos das atividades avaliativas e documentar alterações de cronograma advindas de necessidades identificadas no semestre. O fórum de discussão no ambiente do Moodle será utilizado para intermediar a comunicação entre professor, estagiário de docência, monitor e alunos. Para efeitos da avaliação da participação do aluno na disciplina, as suas estatísticas de utilização da ferramenta de EAD poderão ser levadas em consideração. Como recurso didático auxiliar, o professor poderá empregar vivências na forma de jogos de aprendizado (*serious games*), a serem jogados em equipe ou na forma de objetos de aprendizado digitais, disponibilizados via Moodle, para uso em casa. O sistema Moodle ainda poderá ser utilizado para a entrega e avaliação automatizada, baseada em um conjunto de testes pré-implementados, aplicado aos trabalhos de programação, e de tarefas assinaladas pelo professor. **Pressupostos da metodologia de ensino:** A metodologia adotada nesta disciplina pressupõe que os alunos do curso não se limitem a assistir às aulas, mas que utilizem para as atividades extras associadas a esta disciplina (leituras, resolução de exercícios teóricos e exercícios práticos de programação). Pressupõe-se que os alunos tenham estudado o conteúdo utilizando a bibliografia indicada e tenham resolvido, como atividade extra, todos os exercícios propostos pelo professor.

Controle de frequência

A presença em encontros será registrada pelo professor. O controle de frequência é mantido no Moodle e atualizado após cada dia de aula.

---

6. Avaliação de Aprendizagem

A avaliação regular dos alunos se dará por meio de um conjunto de oito notas, sendo três destas referentes a trabalhos práticos, duas referentes a provas teóricas, duas referentes a provas práticas de programação e uma referente a testes de conhecimento. A média final será composta pela média ponderada destas 8 notas conforme explanado abaixo:

- **N1 (peso 1) -** Média simples de notas atribuídas a pequenos trabalhos de implementação, denominados **Testes Práticos**, para a prática de programação. Serão aproximadamente 10 trabalhos, agendados e entregues no Moodle, de acordo com a avaliação do professor do perfil de aprendizado da turma e suas dificuldades.
- **N2 (peso 1)** - Um projeto de implementação, denominado **Projeto I**, envolvendo um problema do mundo real que possa ser resolvido com a aplicação de pilhas, listas ou filas e ponteiros e alocação dinâmica de memória. Este trabalho poderá ser realizado individualmente ou em dupla e, em havendo necessidade, será concedida uma defesa da implementação.
- **N3 (peso 1)** - Uma prova de prática de programação individual, denominada **Prova Prática I**, realizada em laboratório, cobrindo os assuntos de estruturas lineares. A prova prática será feita utilizando o sistema Moodle, no qual métodos ou aplicações de uma estrutura de dados deverão ser implementados e deverão passar por testes unitários automatizados de acordo com o preceitos utilizados em competições de olimpíadas de programação. A prova incluirá restrições de tempo para a implementação, tempo para a execução e quantidade e técnicas de gerência de memória para a execução.
- **N4 (peso 2)** - Um prova teórica individual, denominada **Prova Teórica I**, cobrindo os assuntos de estruturas lineares e complexidade de algoritmos, realizada em sala de aula e sem consulta, onde serão avaliados aspectos do conhecimento teórico.

- **N5 (peso 1)** - Um Projeto de Implementação, denominado **Projeto II**, envolvendo um problema do mundo real que possa ser resolvido com a aplicação de recuperação eficiente de dados com base em indexação de arquivos. Este trabalho poderá ser realizado individualmente ou em dupla e, em havendo necessidade, será concedida uma defesa da implementação.
- **N6 (peso 1)** - Uma prova de prática de programação individual, denominada **Prova Prática II**, realizada em laboratório, cobrindo os assuntos de estruturas hierárquicas. A prova prática será feita utilizando o sistema Moodle, no qual métodos ou aplicações de uma estrutura de dados deverão ser implementados e deverão passar por testes unitários automatizados de acordo com o preceitos utilizados em competições de olimpíadas de programação. A prova incluirá restrições de tempo para a implementação, tempo para a execução e quantidade e técnicas de gerência de memória para a execução.
- **N7 (peso 2)** - Uma prova teórica individual, denominada **Prova Teórica II**, cobrindo todo o conteúdo da disciplina, realizada em sala de aula e sem consulta, onde serão avaliados aspectos do conhecimento teórico.
- **N8 (peso 1)** - Média simples de notas atribuídas a testes objetivos rápidos, denominados **Testes Teóricos**, a cada aproximadamente 5 itens de ementa, para reforço de aprendizagem teórica.

Portanto, a média final é dada por:

- **MF = (N1 + N2 + N3 + 2 N4 + N5 + N6 + 2 N7 + N8) / 10**

Todos os trabalhos deverão ser entregues juntamente com a documentação exigida pelo professor, sendo a avaliação realizada sobre código e documentação. A forma de entrega é até a data determinada e por meio do Moodle, de acordo com os critérios estabelecidos para cada trabalho, podendo ser em arquivos compactados ou por meio da ferramenta de avaliação automática VPL, dependendo da indicação no enunciado do trabalho. É de responsabilidade do aluno entregar o trabalho na forma correta, arquivos corrompidos ou ilegíveis não serão considerados. Trabalhos entregues após o prazo sofrerão penalizações de 50% da nota independente do atraso. Durante a defesa dos projetos de implementação, o professor se reserva o direito de questionar individualmente os alunos da equipe sobre aspectos teóricos da disciplina contemplados no trabalho, sendo o resultado desses questionamentos levado em consideração de forma individual na atribuição do conceito. Para avaliação dos trabalhos práticos o professor poderá trabalhar com dois modelos de correção: um manual e um automático, sendo dada preferência a este último sempre que possível. Para os casos de de correção automática dos trabalhos de programação, estes serão avaliados por meio da plataforma Moodle. Os testes estarão disponíveis no momento da entrega do trabalho pelos alunos. Os testes serão aplicados e uma nota final do trabalho será sugerida pela ferramenta de correção, podendo o aluno entregar o trabalho quantas vezes forem necessárias para a obtenção da nota desejada, tendo como limite a data de entrega final do trabalho em questão. Cabe ao professor a confirmação da notas em avaliação posterior. Neste caso, a nota pode ser alterada levando em consideração a técnica, o estilo e a eficiência do código à critério do professor e previamente estabelecido no enunciado do trabalho. Os trabalhos corrigidos automaticamente devem passar na sua integralidade e sem quaisquer defeitos nos testes programados para aquela entrega. Para tal os alunos devem sempre testar seus trabalho na plataforma de avaliação, não sendo aceitas reclamações pela não execução no ambiente disponibilizado. Os testes aplicados no processo de avaliação automática podem incluir:

- **Testes baseados na interface de usuários**, onde dados são alimentados para o programa desenvolvido por meio da sua interface e o retorno é avaliado. Nestes casos a precisa implementação da interface é parte da avaliação.
- **Testes unitários em classes**, onde os alunos serão instruídos a implementar uma interface de classe para que as classes entregues possam ser testadas método a método quanto ao seu funcionamento.

- **Avaliação estática de código**, buscando boas práticas de nomeação de atributos, de visibilidade dos métodos, de estruturação e documentação do código.
- **Avaliação dinâmica de código**, buscando boas práticas de gerência de memória e consistência no uso de métodos e classes em tempo de execução.
- **Avaliação de desempenho**, utilizando restrições de tempo e memória para a execução dos trabalhos no sistema de avaliação disponibilizado pela plataforma Moodle.

Os pesos de cada item de avaliação, nos trabalhos corrigidos automaticamente, serão divulgados junto ao enunciado de cada trabalho e podem variar de acordo com o objetivo didático de cada trabalho. No caso de correções manuais de trabalhos que não envolvam a defesa, serão utilizados 5 critérios objetivos, sendo que cada critério vale 2 pontos:

- **Compreensão do Problema:** entendeu o que era para fazer (1 ponto) e modelou corretamente a solução? (1 ponto)
- **Emprego dos Algoritmos e Estruturas:** empregou os algoritmos e estruturas corretos? (% dos algoritmos/estruturas necessários * 2 pontos)
- **Implementação dos Algoritmos e Estruturas:** os implementou corretamente? (% dos algoritmos e estruturas necessários * 2 pontos)
- **Código:** compilou? (1 ponto); funcionou? (1 ponto)
- **Documentação:** código está documentado e programado como indicado no início do semestre? (comentários, 1 ponto; estilo de codificação, 1 ponto)

Havendo defesa dos Projetos de Implementação, a mesma é considerada Prova e o não comparecimento injustificado implica em conceito nulo nesses trabalhos, sendo os critérios de avaliação da defesa os abaixo:

- **Compreensão do Problema:** entendeu o que era para fazer? (2 pontos)
- **Solução:** soube encontrar uma solução? (2 pontos)
- **Conhecimento teórico:** compreendeu as implicações teóricas da solução escolhida? (2 pontos)
- **Algoritmos:** possui compreensão dos algoritmos empregados e sabe descrevê-los? (2 pontos)
- **Código:** compreende a implementação, sabendo detalhar aspectos de seu funcionamento ou de falhas que estão ocorrendo? (2 pontos)

Para efeitos de defesa, os trabalhos práticos dos alunos deverão estar disponíveis nos links de entrega disponibilizados no Moodle da disciplina, não podendo ser trocados ou atualizados quaisquer arquivos. Todas as defesas poderão ser realizadas na presença de uma testemunha (preferencialmente o estagiário de docência, aluno de pós-graduação ou professor do INE) para protocolar as perguntas e respostas. **Plágio e Deslealdade Acadêmica:** Todos as atividades da disciplina seguirão um código ético severo, onde os trabalhos serão avaliados quanto a plágio de forma automatizada por uma ferramenta que o professor dispõe. Caso esta ferramenta indique a possibilidade de plágio, o professor conferirá automaticamente a nota ZERO a todos os envolvidos e, em caso de reincidência, reportará ao órgão competente da universidade para a abertura de processo disciplinar e a aplicação das punições regimentais previstas. **Frequência na Disciplina:** Para a avaliação da frequência na Disciplina fica rigidamente expresso o entendimento disposto na resolução 017/Cun/97 que determina a necessidade de 75% de presença para os cursos presenciais da Universidade. Todos os registros de frequência serão feitos pela ferramenta Moodle e estarão disponíveis para consulta por parte do alunos.

- **Direito de uso de imagem:** fica preservado o direito de imagem do professor e dos alunos participantes das apresentações e aulas, não sendo permitida a utilização das imagens, áudio ou conteúdo para outros fins que não os objetivos pedagógicos, explicitamente definidos para esta disciplina. O uso indevido da imagem de professor e dos alunos é crime previsto na Constituição Federal. Por este motivo não é permitido aos alunos gravar e compartilhar imagens e falas de docentes e discentes.

7. Recuperação

Dado que a disciplina apresenta pelo menos 50% da carga horária consistindo de aulas práticas, conforme deliberação do Colegiado do Curso de Ciências da Computação de 18 de março de 2008, ela não prevê a realização de avaliação no final do semestre (recuperação) de que trata o parágrafo 2º do artigo 70 da Resolução 17/CUn/97.

8. Cronograma

Semana 1 - Apresentação da disciplina. Introdução à programação C++ e ao ambiente de desenvolvimento
Semana 2 - Gerência e alocação dinâmica de memória
Semana 3 - Modelagem e programação de Pilha e Fila em vetor (arrays)
Semana 4 - Lista em vetor (array) como caso geral de Pilha e Fila
Semana 5 - Lista Encadeada
Semana 6 - Fila Encadeada e Pilha Encadeada como especializações de Listas Encadeadas.
Semana 7 - Lista Circular. Enunciado do Projeto I
Semana 8 - Conceito de Complexidade de Algoritmos
Semana 9 - Prova Teórica I (em laboratório) e Prova Prática I (em sala)
Semana 10 - Árvores Binárias de Busca. Árvores Binárias de Busca Semibalanceadas
Semana 11 - Auxílio com trabalhos pendentes. Defesa do Projeto I
Semana 12 - Árvore AVL. Enunciado do Projeto II
Semana 13 - Árvore Rubro-Negra (Red-Black)
Semana 14 - Gerência de Arquivos. Árvores de Busca Semibalanceadas Multivias
Semana 15 - Hashing. Lista invertida
Semana 16 - Métodos de Ordenação - Quicksort e Heapsort.
Semana 17 - Prova Prática II (em laboratório) e Prova Teórica II (em sala)
Semana 18 - Auxílio com trabalhos pendentes. Defesa do Projeto II

9. Bibliografia Básica

[1] DROZDEK, A. Estrutura de Dados e Algoritmos em C++. 4. ed. Cengage Learning, 2017. ISBN: 9788522126651. Disponível em: https://www.cengage.com.br/ls/ebook-estrutura-de-dados-e-algoritmos-em-c/ (acesso via http://portal.bu.ufsc.br/)

[2] MORIN, P. Open Data Structures (in C++). Edition 0.1G?. Disponível em: http://opendatastructures.org/

[3] LUCCHESI, C.L., KOWALTOWSKI, T. Estruturas de Dados e Técnicas de Programação. Instituto de Computação, UNICAMP, 2004. Disponível em: https://www.ic.unicamp.br/~mc202abcd/ln_tkcll.pdf

[4] RICARTE, I.L.M. Estruturas de dados. Faculdade de Engenharia Elétrica e de Computação, UNICAMP, 2008. Disponível em: http://calhau.dca.fee.unicamp.br/wiki/images/0/01/EstruturasDados.pdf

10. Bibliografia Complementar

[1] Materiais disponibilizados via Moodle.

[2] PEREIRA, Silvio do Lago. Estruturas de dados fundamentais: conceitos e aplicações. 12. ed., rev. e atual. São Paulo: Érica, 2009. 238 p. ISBN 9788571943704

[3] JOYANES AGUILAR, Luis. Programação em C++: algoritmos, estruturas de dados e objetos. São Paulo: McGraw Hill, 2008. xxxi, 768 p. ISBN 9788586804816

[4] WIRTH, Niklaus; ZÜRICH, Eth. Algoritmos e estruturas de dados. Rio de Janeiro: LTC, 1999. 255 p. ISBN 978-85-216-1190-5

[5] HOROWITZ, Ellis. Fundamentos de estruturas de dados.. Rio de Janeiro: Ed. Campus, 1986.

[6] CLAYBROOK, Billy G. (Billy Gene). Tecnicas de gerenciamento de arquivos. 2. ed. Rio de Janeiro: Campus, 1987. 248p. ISBN 8570012608

[7] SILBERSCHATZ, KORTH E SUDARSHAN. Database System Concepts. 4ª ed., McGraw-Hill, 2001.

[8] THARP, A. File Organization and Processing. John Wiley, 1988.

[9] SEDGEWICK, R. Algorithms in C++. Addison Wesley, 1996.

[10] GOODRICH, M.T., TAMASSIA, R. Estruturas de Dados e Algoritmos em Java. 4a. Edição. Ed. Bookman, 2007.

[11] AMERAAL, L. Algorithms and Data Structures in C++. Editora John Wiley, 1996.

[12] TENNENBAUM, A., LANGSAM, Y. Data Structures using C and C++. 2ª Ed. Prentice-Hall, 1995.