



SERVIÇO PÚBLICO FEDERAL
MINISTÉRIO DA EDUCAÇÃO
UNIVERSIDADE FEDERAL DE SANTA CATARINA
CENTRO TECNOLÓGICO
DEPARTAMENTO DE ENGENHARIA DO CONHECIMENTO
CAMPUS UNIVERSITÁRIO SEDE TRINDADE
CEP: 88040-900, FLORIANÓPOLIS - SC, Brasil.

PLANO DE ENSINO

1. IDENTIFICAÇÃO:

Disciplina:	Programação II				
Código da disciplina:	EGC5301				
Carga horária semanal:	6 h/a				
Carga horária total e distribuição dela:	108 h/a (em horas-aula)	Teórica:	54 h/a	PCC:	108 h/a
		Prática:	54 h/a	Extensão:	0 h/a
Curso:	Curso Superior de Tecnologia em Ciência de Dados				
Pré-requisito:	CIN8001 - Programação I				
Fase e turma da oferta:	2a fase, turma 02349				
Semestre da oferta:	2026/02				
Horário da aula:	Quarta-feira: 14:20 -17:40 Labtec (sala 202, 2o. andar do Bloco D do CED) Sexta-feira: 14:20 - 16:00 Labtec (sala 202, 2o. andar do Bloco D do CED)				
Professor/a:	Prof. Dr. Giovani Gracioli				
Contato:	giovani.g@ufsc.br				
Horário de atendimento de estudantes:	Quarta-feira, 10 às 12 hrs				

2. EMENTA:

Introdução à programação orientada a objetos. Fundamentos da orientação a objetos: herança, herança múltipla, polimorfismo, encapsulamento, construtores e destrutores, associação. Sobrecarga de operadores. Introdução às estruturas de dados. Pilhas, filas, listas, árvores, tuplas, dicionários e sets. Algoritmos de ordenação e busca.

3. OBJETIVOS:

3.1 Geral:

Entender os conceitos relativos aos fundamentos da orientação a objetos, ao uso e implementação de estruturas de dados e algoritmos de ordenação e busca, a fim de possibilitar a implementação de programas computacionais avançados.

3.2 Específicos:

- a. compreender os conceitos relacionados ao paradigma de programação orientada a objetos;
- b. estruturar programas computacionais utilizando estrutura de dados;
- c. entender e aplicar algoritmos de ordenação e busca;
- d. desenvolver programas computacionais eficientes.

4. CONTEÚDO PROGRAMÁTICO:

Unidade I – Fundamentos da orientação a objetos

- 1.1. classes e objetos;
- 1.2. herança e herança múltipla;
- 1.3. polimorfismo;
- 1.4. encapsulamento;
- 1.5. construtores e destrutores;
- 1.6. associação; e
- 1.7. sobrecarga de operadores;

Unidade II – Fundamentos das estruturas de dados

- 2.1. Pilhas: implementações e aplicações;
- 2.2. Filas: implementações e aplicações;
- 2.3. Listas: implementações e aplicações;
- 2.4. Árvores: implementações e aplicações; e
- 2.5. Tuplas, dicionários e sets.

Unidade III – Algoritmos de ordenação e busca

- 3.1. Complexidade de algoritmos;
- 3.2. Algoritmos de ordenação; e
- 3.3. Algoritmos de busca.

Unidade IV – Prática de programação

5. METODOLOGIA

5.1 Visão Geral

As aulas teóricas consistem em apresentações orais do conteúdo por parte do professor e discussões coletivas dos conteúdos contidos em artigos, capítulos de livros e sites da Internet indicados pelo professor. As aulas práticas envolvem a análise de programas previamente elaborados pelo professor e o desenvolvimento de programas conforme os tópicos do conteúdo programático. Alguns tópicos serão abordados ou aprofundados na forma relatório ou sugestão de estudo. Todo o material de ensino é disponibilizado no Moodle.

5.2 Estagiário de Docência

Além do professor, responsável pela disciplina, está prevista a participação de um estagiário de docência, que irá trabalhar juntamente com o professor no desenvolvimento e avaliação das atividades. As informações sobre o estagiário de docência e sua disponibilidade para atendimentos são disponibilizadas no Moodle.

6. AVALIAÇÃO

6.1 Instrumentos de Avaliação

A disciplina possui as seguintes atividades avaliativas:

- **Provas regulares (P):** Duas provas individuais sem consulta;
- **Trabalhos de implementação (T):** Dois trabalhos práticos de programação com prazo de entrega definido pelo professor;
- **Atividades de fixação (AF):** Diversas atividades de fixação de conteúdo com prazos para entrega definidos pelo professor. Estas atividades compreendem questionários sobre partes do conteúdo, estudos dirigidos ou atividades de sala de aula invertida.
- **Laboratórios Virtuais de Programação (LVP):** Diversas atividades práticas de programação fornecidas como laboratórios virtuais de programação no Moodle.

Os seguintes critérios serão observados para fins de avaliação individual dos(as) discentes:

- Compreensão dos conteúdos discutidos;
- Clareza e qualidade das soluções das atividades práticas;
- Desempenho na apresentação da solução e/ou qualidade do relatório da atividade avaliativa entregue;
- Cumprimento de prazos de entrega das atividades e trabalho prático (não serão aceitas entregas de atividades fora de prazo ou por email).

As atividades avaliativas (Ps, Ts, Afs e LVPs) serão individuais, exceto quando claramente definido no enunciado das mesmas. Além das soluções das atividades, o professor poderá solicitar aos(as) discentes: (i) a entrega de um relatório descritivo com detalhamento da solução proposta e resultados observados ou (ii) apresentação da solução proposta de forma presencial. A entrega e o acompanhamento do trabalho (T), das atividades práticas (LVPs) e das atividades de fixação (AFs) será realizado Moodle.

ATENÇÃO: Não serão toleradas cópias de soluções de atividades avaliativas. No caso de identificação de cópia, todos(as) os(as) discentes envolvidos(as) receberão nota zero. Em caso de reincidência, a situação será levada à coordenação do curso. Ressalta-se que os(as) discentes não deverão fazer uso de repositórios públicos para o desenvolvimento das atividades avaliativas sob o risco de colegas copiarem as soluções disponíveis publicamente. Ao disponibilizar a solução em repositório público os(as) discentes assumem o risco de possibilidade de cópia e, portanto, estes casos também serão considerados como plágio e todos(as) os(as) envolvidos(as) receberão nota zero. Da mesma forma, não serão tolerados o uso de IA para geração de código substituindo parcial ou totalmente a responsabilidade do aluno em termos de codificação. O uso de IA nos trabalhos e atividades deverá ser explicitamente descrito quando houver. O professor irá disponibilizar no moodle um guia de boas práticas para o uso de IA.

6.2 Reposição de Atividades Avaliativas

No caso de impossibilidade de realização de prova (P) por motivo justificável, devidamente comprovado, o(a) discente deverá solicitar ao professor, dentro do prazo de 3 dias úteis, a autorização para substituí-la, através de uma prova substitutiva (PS), cujo conteúdo avaliado será equivalente ao conteúdo total da disciplina. A PS será realizada posteriormente em data definida no cronograma da disciplina. A nota da(s) prova(s) perdida(s) será substituída pela nota obtida na PS.

6.3 Cálculo da Média Final (MF)

Os(as) discentes com frequência suficiente (FS) terão sua média final (MF) calculada da seguinte forma:

$$MF = 0.4 \times (P1 + P2) / 2 + 0.3 \times (T1 + T2) / 2 + 0.25 \times (LVP1 + LVP2 \dots + LVPn) / n + 0.5 \times (AF1 + AF2 + \dots + AFn) / n$$

6.4 Prova de Recuperação (REC)

Conforme parágrafo 2º do artigo 70 da Resolução 17/CUn/97, o aluno com frequência suficiente (FS) e média final no período (MF) entre 3,0 e 5,5 terá direito a uma nova avaliação ao final do semestre (REC), sendo a nota final (NF) calculada conforme parágrafo 3º do artigo 71 desta resolução, ou seja: $NF = (MF + REC) / 2$.

7. CRONOGRAMA

O cronograma da disciplina será disponibilizado no Moodle. O conteúdo programático será dividido nas suas 3 unidades e o cronograma indicará a quantidade de semanas necessárias para cobrir o conteúdo de cada unidade. A prática de programação será realizada dentro de cada unidade. As AFs e LVPs será disponibilizadas no moodle.

As datas aproximadas para a realização das provas são:

- **Provas regulares (P):** 8a e 15a semanas do calendário acadêmico.
- **Prova substitutiva (PS):** Última semana do calendário acadêmico.
- **Prova de recuperação (REC):** Última semana do calendário acadêmico.

8. BIBLIOGRAFIA

8.1 Bibliografia Básica

GUIMARÃES, Angelo de Moura; LAGES, Newton Alberto de Castilho. Algoritmos e estruturas de dados. Rio de Janeiro: Livros Tecnicos e Cientificos, 1985. 216p. ISBN 8521603789.

PREISS, Bruno R. Estruturas de dados e algoritmos: padrões de projetos orientados a objetos com java. Rio de Janeiro: Campus, c2001. xvi, 566p. ISBN 8535206930.

PILGRIM, Mark. Dive Into Python 3. 2nd ed. 2009 .Berkeley, CA. Disponível em: <https://link.springer.com/book/10.1007/978-1-4302-2416-7>. Acesso em: 08 maio 2025.

Ao longo do semestre, a bibliografia é acrescida com a indicação de artigos publicados em periódicos científicos e disponibilizados aos alunos, via Portal de Periódicos da CAPES.

8.2 Bibliografia Complementar

SUMMERFIELD, Mark. Programação em Python 3: Uma Introdução Completa à Linguagem Python. Rio de Janeiro: Alta Books, 2013. 506 p.

GAMMA, E. et al. Design Patterns – Elements of Reusable Object-Oriented Software. AddisonWesley, 1995.

WAZLAWICK, Raul S. Introdução a Algoritmos e Programação com Python. São Paulo: Elsevier, 2018.

