Lógica de Programação

Bem-vindo à disciplina que forma a base do pensamento computacional. Aqui você aprenderá a traduzir problemas reais em soluções estruturadas, desenvolvendo o raciocínio lógico essencial para qualquer programador.

PROF. CLOVES ROCHA

GRADUAÇÃO

Ementa da Disciplina

Esta disciplina oferece uma introdução sólida e progressiva à lógica de programação, cobrindo desde os conceitos fundamentais até estruturas de controle avançadas. O objetivo é capacitar o estudante a pensar algoritmicamente e resolver problemas computacionais de forma estruturada e eficiente.

Conceitos Fundamentais

Estudo da lógica por trás dos programas de computador, tipos de programação e uso de variáveis e operadores aritméticos como alicerce do desenvolvimento.

Linguagem de Alto Nível

Introdução à linguagem de programação, construção de algoritmos, e uso de operadores relacionais e lógicos para expressar condições e comparações.

Estruturas de Controle

Expressões lógicas aplicadas a estruturas de decisão, controle e repetição — elementos que definem o fluxo de execução de qualquer programa.

Funções e Recursividade

Conceito de função como bloco reutilizável de código, e introdução à recursividade como forma elegante de resolver problemas que se repetem em escala menor.

Competências que Você Vai Desenvolver

Ao final da disciplina, espera-se que o estudante tenha desenvolvido um conjunto de competências práticas e teóricas que o habilitam a criar soluções computacionais completas. Estas competências foram cuidadosamente selecionadas para garantir uma formação técnica sólida e alinhada ao mercado.

Dominar Métodos de Algoritmos

Conhecer e aplicar os principais métodos de desenvolvimento de algoritmos, desde fluxogramas até pseudocódigo, compreendendo como estruturar o raciocínio lógico antes mesmo de escrever qualquer código.

Conhecer Estruturas de Programação

Compreender as estruturas fundamentais que compõem qualquer programa — sequência, decisão, repetição e modularização — e saber quando e como aplicar cada uma delas adequadamente.

Criar Projetos Computacionais

Ser capaz de analisar um problema do mundo real, decompô-lo em partes menores e criar um projeto computacional estruturado para resolvê-lo utilizando as ferramentas e estruturas aprendidas ao longo da disciplina.

Unidade I — Conceitos e Lógica de Programas

A primeira unidade estabelece o vocabulário e os conceitos fundamentais que sustentam toda a disciplina. Compreender **por que programamos** e **como os computadores processam instruções** é o primeiro passo para se tornar um desenvolvedor competente.

Importância da Programação

A programação está presente em praticamente todos os aspectos da vida moderna — de aplicativos bancários a sistemas de saúde. Entender sua lógica permite ao profissional de computação criar soluções inovadoras e automatizar processos.

Tipos de Programação

- **Desktop**: aplicações instaladas localmente no computador
- **Web**: sistemas acessados via navegador na internet
- **Mobile**: apps para dispositivos móveis (Android, iOS)
- **Embarcado**: software integrado a hardware específico (microcontroladores, sensores)

Variáveis na Programação

Uma variável é um espaço reservado na memória do computador para armazenar um valor que pode mudar durante a execução do programa. Cada variável possui um nome, um tipo de dado e um valor. Compreender variáveis é essencial para manipular informações dentro de um algoritmo.

Operadores Aritméticos

- **+** Adição — soma dois valores
- **-** Subtração — diferença entre valores
- ***** Multiplicação — produto de dois valores
- **/** Divisão — quociente entre valores
- **%** Módulo — resto da divisão inteira

Unidade II — Lógica com Linguagem de Alto Nível

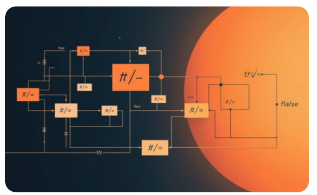
Nesta unidade, o estudante começa a trabalhar com uma linguagem de programação real, aplicando os conceitos de algoritmos e explorando os operadores que permitem comparar valores e combinar condições lógicas. Esta é a ponte entre o raciocínio abstrato e o código executável.

Algoritmos

Um algoritmo é uma sequência finita, ordenada e não ambígua de passos para resolver um problema. Antes de escrever código, o bom programador estrutura sua solução em pseudocódigo ou fluxograma, garantindo que a lógica esteja correta independentemente da linguagem usada.

Operadores Relacionais

Permitem comparar dois valores e retornam verdadeiro ou falso: **!=** (diferente), **==** (igual), **>=** (maior ou igual), **<=** (menor ou igual), **>** (maior que), **<** (menor que). São a base das decisões nos programas.



Operadores Lógicos

Combinam expressões booleanas: **E (AND)** — verdadeiro somente se ambas as condições forem verdadeiras; **OU (OR)** — verdadeiro se ao menos uma for verdadeira; **NEGAÇÃO (NOT)** — inverte o resultado da expressão. Fundamentais para criar condições complexas.

Unidade III — Estruturas de Controle e Repetição

O poder real da programação reside na capacidade de tomar decisões e repetir ações automaticamente. Esta unidade apresenta as três grandes estruturas que controlam o fluxo de qualquer programa, sendo indispensáveis para criar soluções dinâmicas e eficientes.

1

Estruturas de Decisão

Permitem que o programa execute blocos diferentes de código dependendo de uma condição. As principais são **if/else** (se/senão) e **switch/case**. Por exemplo: se o usuário digitar uma senha correta, acesse o sistema; senão, exiba um erro.

2

Estruturas de Controle

Determinam a ordem em que as instruções são executadas. Incluem comandos como **break** (interrompe um laço), **continue** (pula para a próxima iteração) e **return** (encerra uma função e retorna um valor). Controlam com precisão o fluxo do programa.

3

Estruturas de Repetição

Permitem executar um bloco de código múltiplas vezes. Os principais laços são **for** (quando se sabe o número de repetições), **while** (enquanto uma condição for verdadeira) e **do-while** (executa ao menos uma vez antes de checar a condição).

Unidade IV — Funções e Recursividade

Funções são a espinha dorsal da programação moderna. Elas permitem dividir um programa complexo em partes menores, reutilizáveis e testáveis de forma independente. A recursividade, por sua vez, é uma técnica elegante e poderosa que representa o ápice do pensamento algorítmico nesta disciplina.

Conceito de Função

Uma função é um bloco de código nomeado que realiza uma tarefa específica e pode ser chamado (invocado) várias vezes ao longo do programa. Toda função pode receber **parâmetros** (dados de entrada) e retornar um **valor de saída**. O uso de funções promove:

- **Reutilização de código** — escreva uma vez, use várias vezes
- **Legibilidade** — código organizado e com nomes descritivos
- **Manutenção** — alterações feitas em um único lugar se propagam para todo o programa
- **Abstração** — esconde a complexidade por trás de uma interface simples

Lógica da Recursividade

Uma função recursiva é aquela que **chama a si mesma** durante sua execução. Para funcionar corretamente, toda função recursiva precisa de dois elementos essenciais:

- **Caso base**: a condição que encerra as chamadas recursivas, evitando loop infinito
- **Caso recursivo**: a parte que reduz o problema e chama a função novamente

Exemplos clássicos: cálculo de fatorial (n! = n × (n-1)!), sequência de Fibonacci, busca binária e travessia de árvores. A recursividade transforma problemas complexos em soluções elegantes e concisas.

Conteúdo Programático — Linha do Tempo

Visualize a progressão da disciplina ao longo do semestre. Cada unidade foi cuidadosamente ordenada para construir o conhecimento de forma gradual e consistente — dos conceitos mais simples até os mais abstratos. Respeitar essa progressão é fundamental para absorver bem os conteúdos.

1 Unidade I

Conceitos Fundamentais

- Importância da programação
- Tipos: desktop, web, mobile, embarcado
- Variáveis e tipos de dados
- Operadores aritméticos

2 Unidade II

Linguagem de Alto Nível

- Introdução à linguagem
- Construção de algoritmos
- Operadores relacionais
- Operadores lógicos (E, OU, NÃO)

3 Unidade III

Expressões e Estruturas

- Estruturas de decisão (if/else)
- Estruturas de controle (break/continue)
- Estruturas de repetição (for/while)

4 Unidade IV

Funções e Recursividade

- Definição e chamada de funções
- Parâmetros e retorno
- Conceito de recursividade
- Casos base e recursivos

A disciplina é estruturada em **quatro unidades progressivas**, onde cada etapa constrói sobre a anterior. Alunos que acompanham o ritmo proposto tendem a ter melhor desempenho nas avaliações e maior facilidade nas disciplinas subsequentes.

Metodologia de Ensino e Aprendizagem

A disciplina adota uma abordagem ativa e mediada, combinando teoria e prática para garantir que o estudante não apenas compreenda os conceitos, mas saiba aplicá-los em situações reais. As atividades são realizadas de forma síncrona e presencial, com o professor atuando como mediador entre o estudante e o conhecimento.

Atividades Teóricas

Aulas expositivas e dialogadas para apresentação dos conceitos, com incentivo à participação ativa dos estudantes por meio de perguntas, discussões e exemplos do cotidiano tecnológico. O objetivo é construir uma base conceitual sólida antes da prática.

Atividades Práticas

Exercícios de programação aplicados diretamente no computador, com resolução de problemas reais utilizando a linguagem de alto nível estudada. A prática reforça os conceitos teóricos e desenvolve habilidades de depuração e raciocínio lógico.

Ferramentas e Estratégias

Uso de diferentes recursos didáticos, como ambientes de programação online, desafios de lógica, pseudocódigo, fluxogramas e projetos em grupo. A diversidade de estratégias garante diferentes formas de aprendizado para diferentes perfis de alunos.

Sistema de Avaliação e Critérios de Aprovação

A avaliação é um instrumento formativo e somativo, incidindo sobre **frequência** (mínimo de 75% de presença obrigatória) e **aproveitamento** (notas de 0 a 10). Serão realizadas no mínimo 2 avaliações parciais por período. Conheça os critérios completos abaixo:

Composição das Notas

A **Média Parcial (MP)** é calculada pela média aritmética simples das avaliações parciais realizadas durante o semestre. As avaliações podem incluir provas tradicionais, Prova Colegiada, trabalhos, exercícios e outras atividades definidas pelo professor com aprovação da coordenadoria.

Critérios de Aprovação — Média Parcial

- **MP ≥ 7,0** → APROVADO diretamente
- **4,0 ≤ MP < 7,0** → AVALIAÇÃO FINAL obrigatória
- **MP < 4,0** → REPROVADO sem direito a final

Critérios de Aprovação — Média Final

Caso o aluno realize a Avaliação Final, a **Média Final (MF)** é calculada pela média aritmética simples entre a Média Parcial e a nota da Avaliação Final:

- **MF ≥ 5,0** → APROVADO
- **MF < 5,0** → REPROVADO

Resumo Visual — Fluxo de Aprovação

01

Realizar as Avaliações Parciais

Participar de todas as avaliações ao longo do semestre. Mínimo de 2 avaliações obrigatórias para compor a Média Parcial.

02

Calcular a Média Parcial

Média aritmética simples de todas as avaliações parciais realizadas. MP ≥ 7,0 garante aprovação direta.

03

Avaliação Final (se necessário)

Obrigatória para alunos com 4,0 ≤ MP < 7,0. A nota final será a média entre a MP e a nota da prova final.

04

Verificar Frequência

Independente das notas, a frequência mínima de 75% da carga horária é obrigatória para não ser reprovado por falta.

Dica do Professor: Mantenha presença constante nas aulas e entregue todas as atividades no prazo. A consistência ao longo do semestre é o caminho mais seguro para a aprovação direta com média parcial ≥ 7,0.