AULA 2 — A Importância da Programação

Prof. Cloves Rocha · ADS & Ciência da Computação

Nesta aula exploramos por que a programação é uma habilidade central no mundo contemporâneo. Desde aplicativos bancários até sistemas de saúde, a lógica de programação permite criar automações, resolver problemas complexos e transformar ideias em produtos reais. Abaixo, apresentamos um panorama dos principais tipos de programação, exemplos de aplicação e uma atividade prática para você exercitar lógica e raciocínio.

VISÃO GERAL

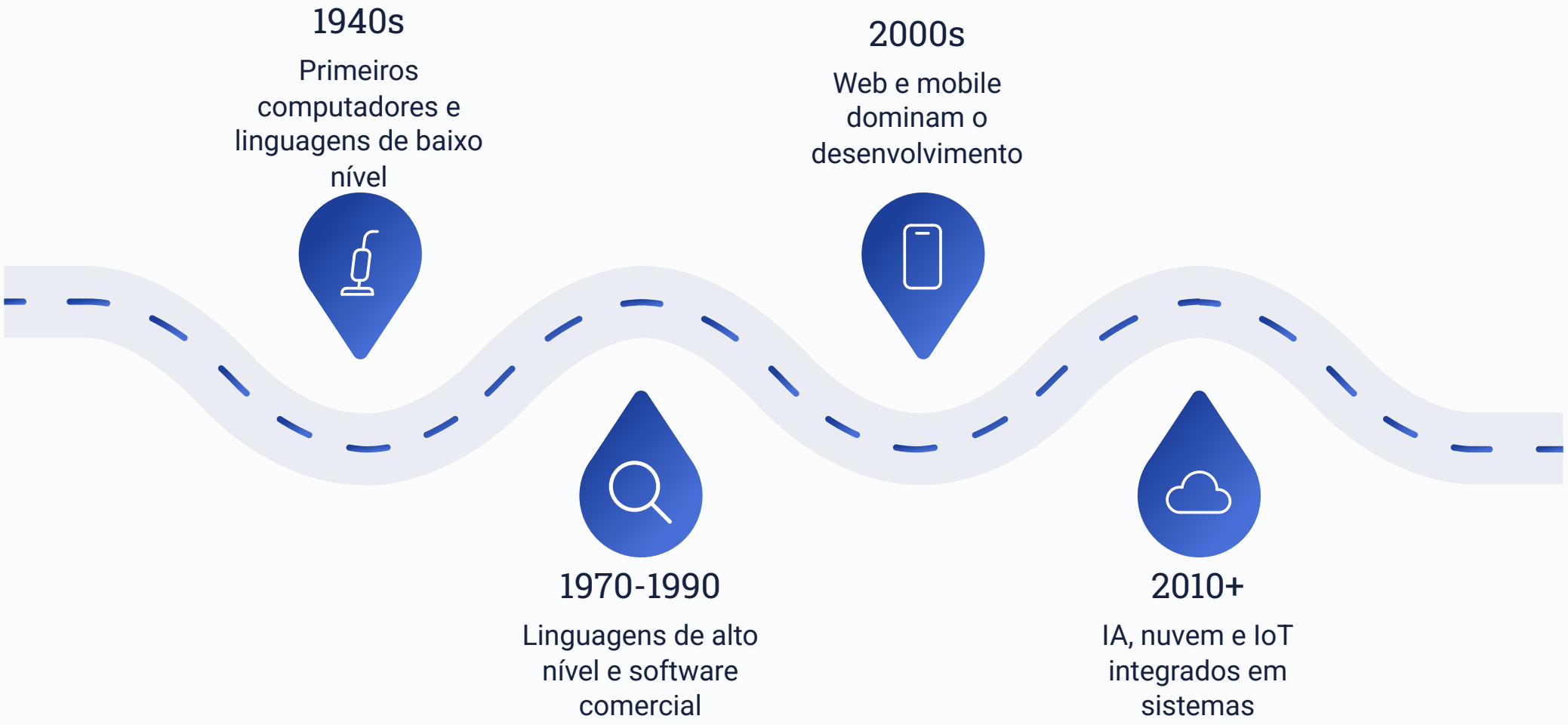
Por que aprender programação?

Competência essencial

Programação é uma competência requisitada em quase todos os setores: finanças, saúde, indústria, educação e entretenimento. Saber programar amplia suas oportunidades de carreira.

Além da empregabilidade, programar desenvolve pensamento lógico, capacidade de decompor problemas e criar soluções reutilizáveis — habilidades fundamentais para qualquer profissional técnico.

Linha do tempo: evolução e impacto da programação



Essa linha do tempo ilustra como a prática da programação se transformou: de rotinas de baixo nível em máquinas grandes até plataformas em nuvem, dispositivos móveis e sistemas inteligentes. Entender essa trajetória ajuda a contextualizar as opções de carreira e as tecnologias emergentes.

Tipos principais de programação

Desktop

Aplicações instaladas localmente no computador: suítes de escritório, softwares de edição, ferramentas CAD e aplicações industriais. Características: acesso direto ao hardware, bom desempenho, instalação por usuário ou por máquina.

Web

Sistemas acessados via navegador: sites, portais, aplicações de e-commerce e plataformas SaaS. Características: multiplataforma, deployment contínuo, integração com APIs e bancos de dados.

Mobile

Aplicativos para Android e iOS. Foco em experiência do usuário, interação por toque, consumo de sensores e otimização de recursos. Pode ser nativo, híbrido ou progressivo (PWA).

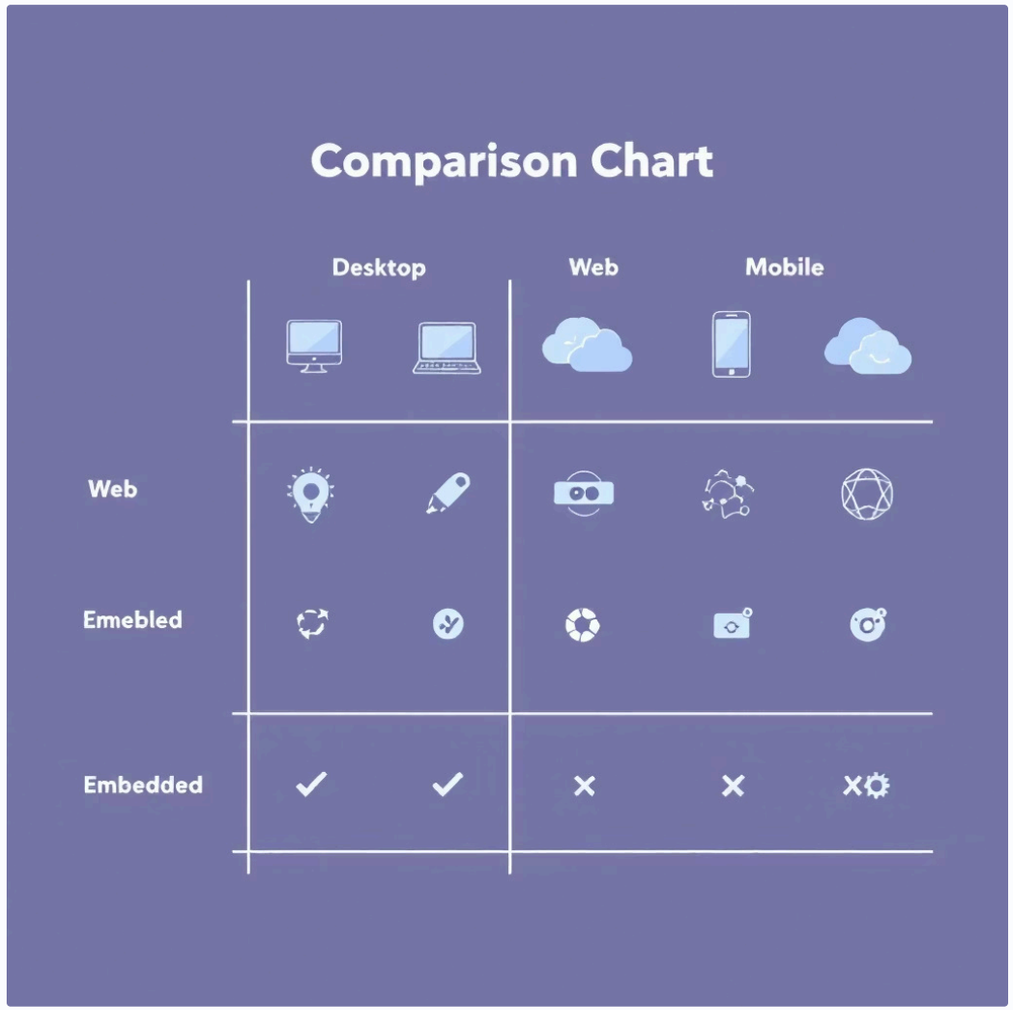
Embarcado

Software integrado a hardware: microcontroladores, IoT e sistemas críticos. Características: restrição de memória, necessidade de eficiência, comunicação com sensores e atuadores. Muito usado em automação e produtos conectados.

Comparando os tipos: quando escolher cada um

Critérios de seleção

- Alcance do usuário (local vs. global)
- Requisitos de desempenho (tempo real vs. batch)
- Complexidade de interface (UI/UX avançada vs. simples)
- Integração com hardware (necessária em embarcados)
- Manutenção e atualizações (web facilita deploy contínuo)



Resumindo: escolha desktop para aplicações críticas de alto desempenho; web para alcance e atualizações rápidas; mobile para experiência rica em dispositivos pessoais; embarcado quando o software controla hardware físico.

Exemplos de aplicação no mundo real

Setor financeiro

Sistemas de transações, análise de risco e antifraude que dependem de programação segura e escalável.

Saúde

Prontuários eletrônicos, agendamento e monitoramento remoto que salvam vidas com integração de dados.

Indústria e automação

Controle de linhas de produção, sensores e robôs — domínio de embarcados e comunicação M2M.

Comércio eletrônico

Plataformas web e mobile que conectam vendedores e consumidores, com sistemas de pagamento e logística.

Habilidades e ferramentas recomendadas

Lógica e algoritmos

Fundamento para resolver problemas, escrever pseudocódigo e transformar requisitos em passos lógicos.

Linguagens básicas

Comece por Python (versátil), JavaScript (web) e C/C++ (embarcado/alto desempenho).

Ferramentas

IDEs (VS Code), controle de versão (Git), ambientes de deploy (Docker, servidores web).

Boas práticas

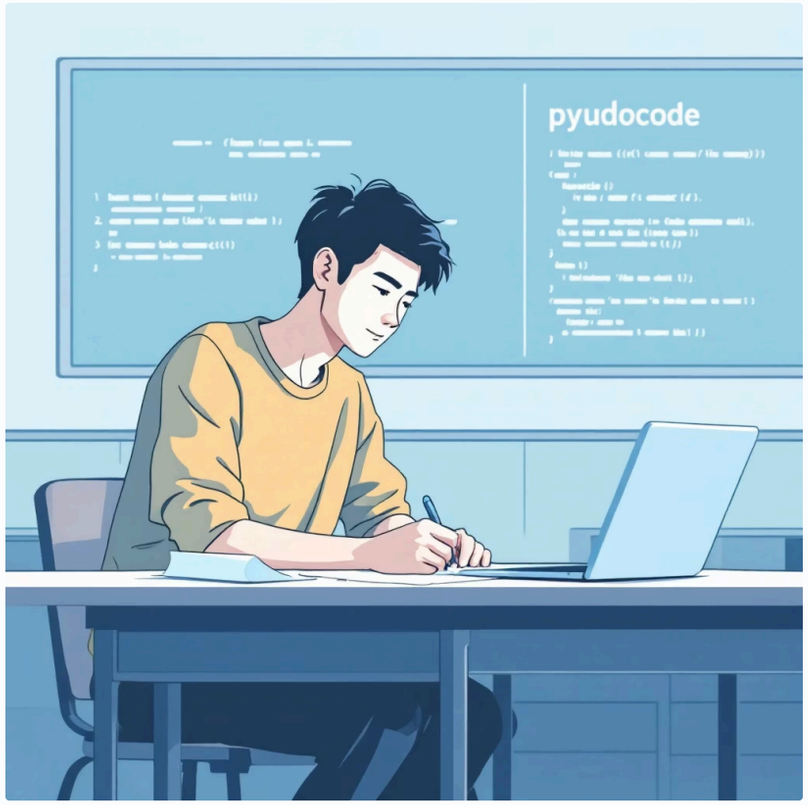
Testes automatizados, documentação, versionamento e revisão por pares.

Desenvolver essas habilidades vai facilitar sua transição entre áreas (web, mobile, embarcado) e tornar seu perfil mais competitivo no mercado.

Atividade prática — PRÁTICA 1: Exercício individual

Objetivo: praticar lógica de programação resolvendo um problema simples em pseudocódigo ou em uma linguagem que você conhece. Tempo estimado: 30–45 minutos.

1. **Problema**: Crie um algoritmo que receba a idade de várias pessoas e retorne: maior idade, menor idade, média de idades e quantas pessoas são maiores de 18 anos.
2. **Requisitos**: leia pelo menos 5 idades (entrada), valide entradas (idade > 0), use estruturas de repetição e condicionais.
3. **Entregável**: código funcional + breve explicação (3–5 linhas) sobre a lógica usada.



Sugestão de solução (em pseudocódigo)

Iniciar contador = 0, soma = 0, maior = -inf, menor = +inf, maiores18 = 0. Enquanto contador < N: ler idade; validar; atualizar soma, maior, menor; se idade >= 18 então maiores18++. Ao final, média = soma / N; imprimir maior, menor, média, maiores18.

Dicas para estudar programação de forma eficiente

Pratique regularmente

Consistência supera intensidade: pequenos exercícios diários consolidam a lógica e a sintaxe.

Aprenda fazendo projetos

Construa mini-projetos (to-do list, calculadora, monitor de sensores) para aplicar teoria em contexto real.

Estude em comunidade

Troque feedback, participe de grupos de estudo e code reviews — isso acelera seu aprendizado.

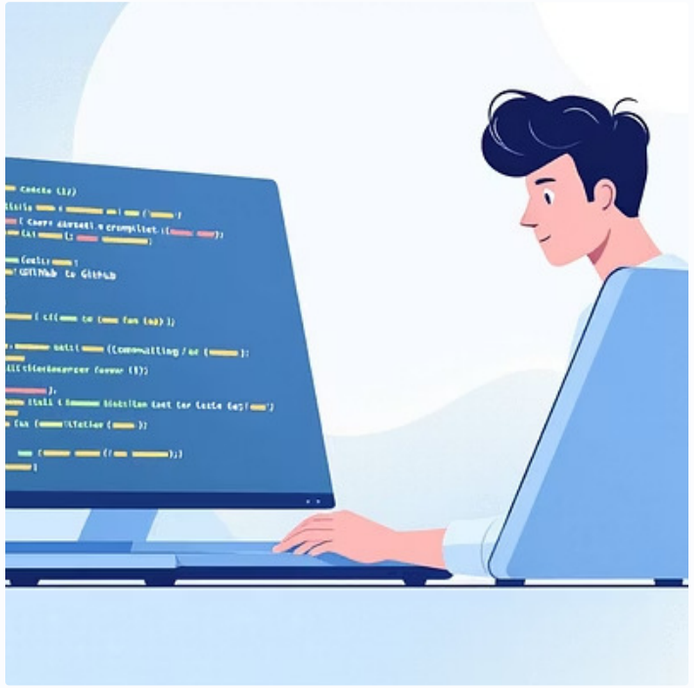
Leia e escreva código

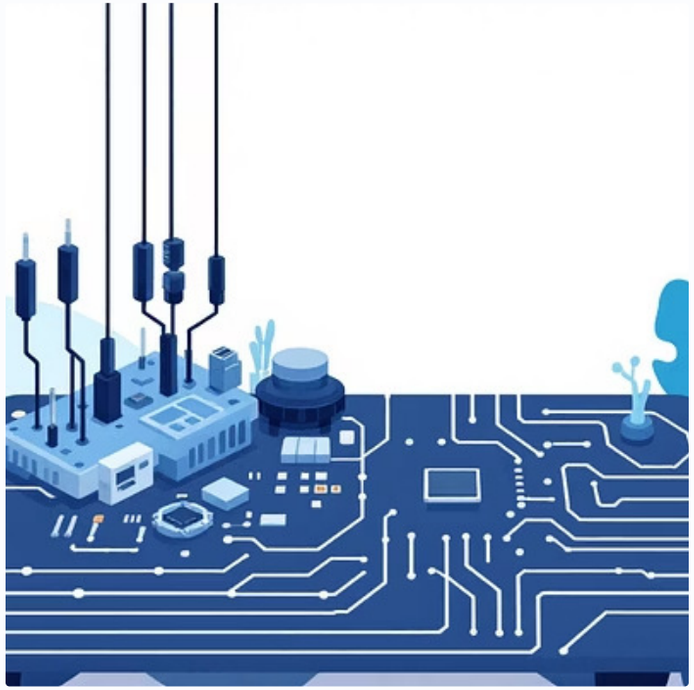
Ler código alheio e escrever suas próprias soluções desenvolve padrões mentais e boas práticas.

Resumo e próximos passos

A programação é uma ferramenta poderosa que permite automatizar tarefas, construir produtos e transformar processos. Nesta aula você: entendeu os tipos (desktop, web, mobile, embarcado), viu exemplos reais, recebeu recomendações de ferramentas e realizou uma prática de lógica. Próximos passos sugeridos:

- Completar a PRÁTICA 1 e subir o código para um repositório Git.
- Experimentar um mini-projeto relacionado ao seu interesse (web, mobile ou embarcado).
- Rever a linha do tempo para identificar tecnologias que você quer aprender nos próximos meses.





Dica: Use a cor tema #3257b8ff como destaque em seus projetos e anotações para criar uma identidade visual consistente.