

CÓDIGO E CAFÉ

Introdução ao Pensamento Computacional



Adriana Oliveira

$f(x)$ Você já se perguntou como os computadores conseguem resolver problemas complexos?

Neste eBook, vamos explorar o que é o pensamento computacional e como você pode começar a desenvolvê-lo.

Então, prepare-se para embarcar em uma jornada emocionante rumo ao mundo da lógica, da criatividade e da resolução de problemas.

BOAS-VINDAS AO MUNDO DO PENSAMENTO COMPUTACIONAL !

< “Bons estudos!” >



Pensamento computacional é como um superpoder que te ajuda a resolver problemas de uma maneira que um computador entende o que fazer.

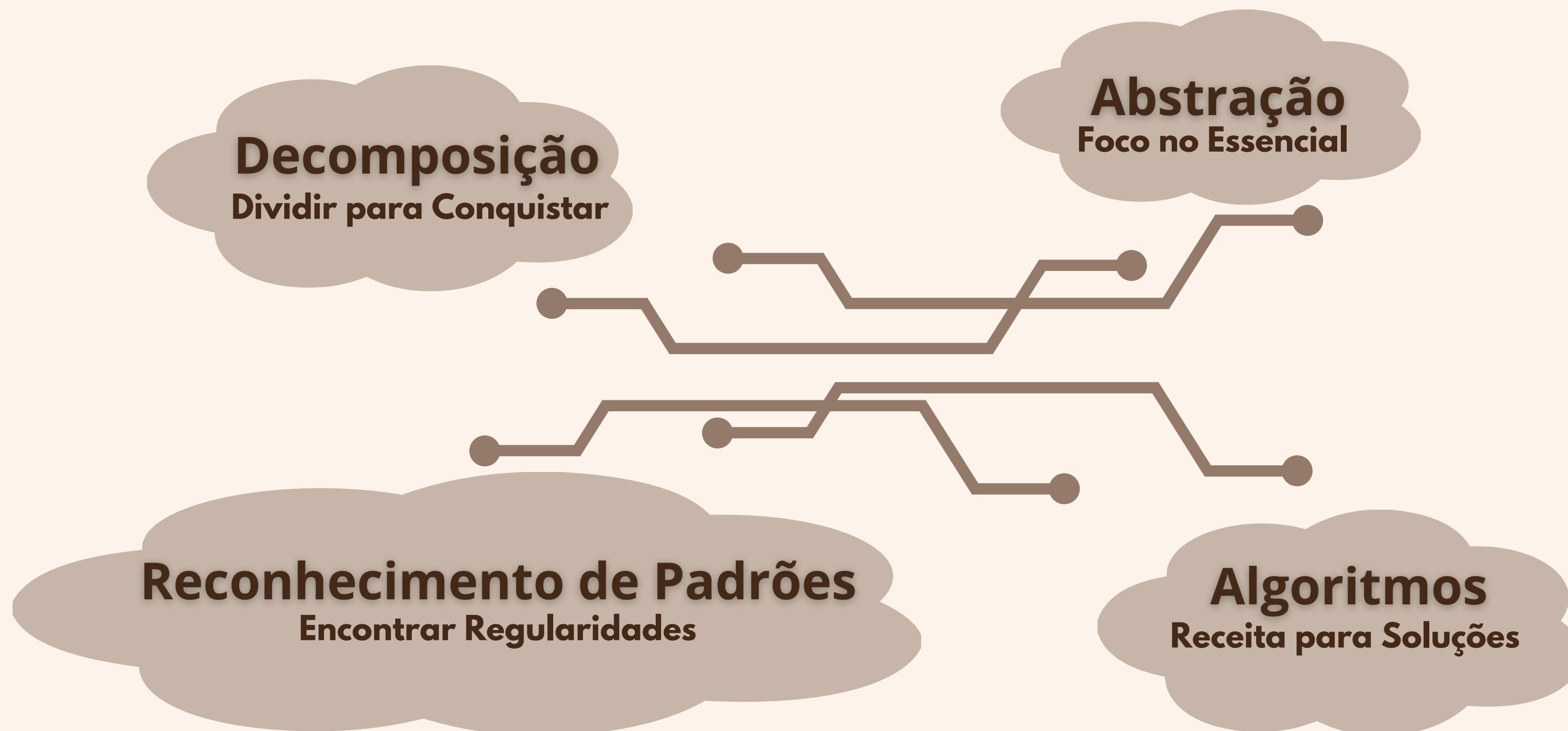
É uma habilidade importante na ciência da computação, mas não se limita apenas à programação. Pode ser útil em muitas situações do dia a dia, te ajudando a resolver problemas de uma forma mais organizada e inteligente.

É como se você quebrasse um problema grande em partes menores, procurasse por padrões, tirasse as partes que não são importantes e criasse um plano eficiente para resolver o problema.

Em resumo, o pensamento computacional é fundamental no mundo da programação, e também te capacita a resolver uma variedade de problemas de forma mais eficiente e eficaz.

◆ Aqui estão os princípios do pensamento computacional.

Eles capacitam as pessoas para abordar problemas de forma estruturada, criativa e eficiente, tanto na programação quanto em outras áreas da vida.

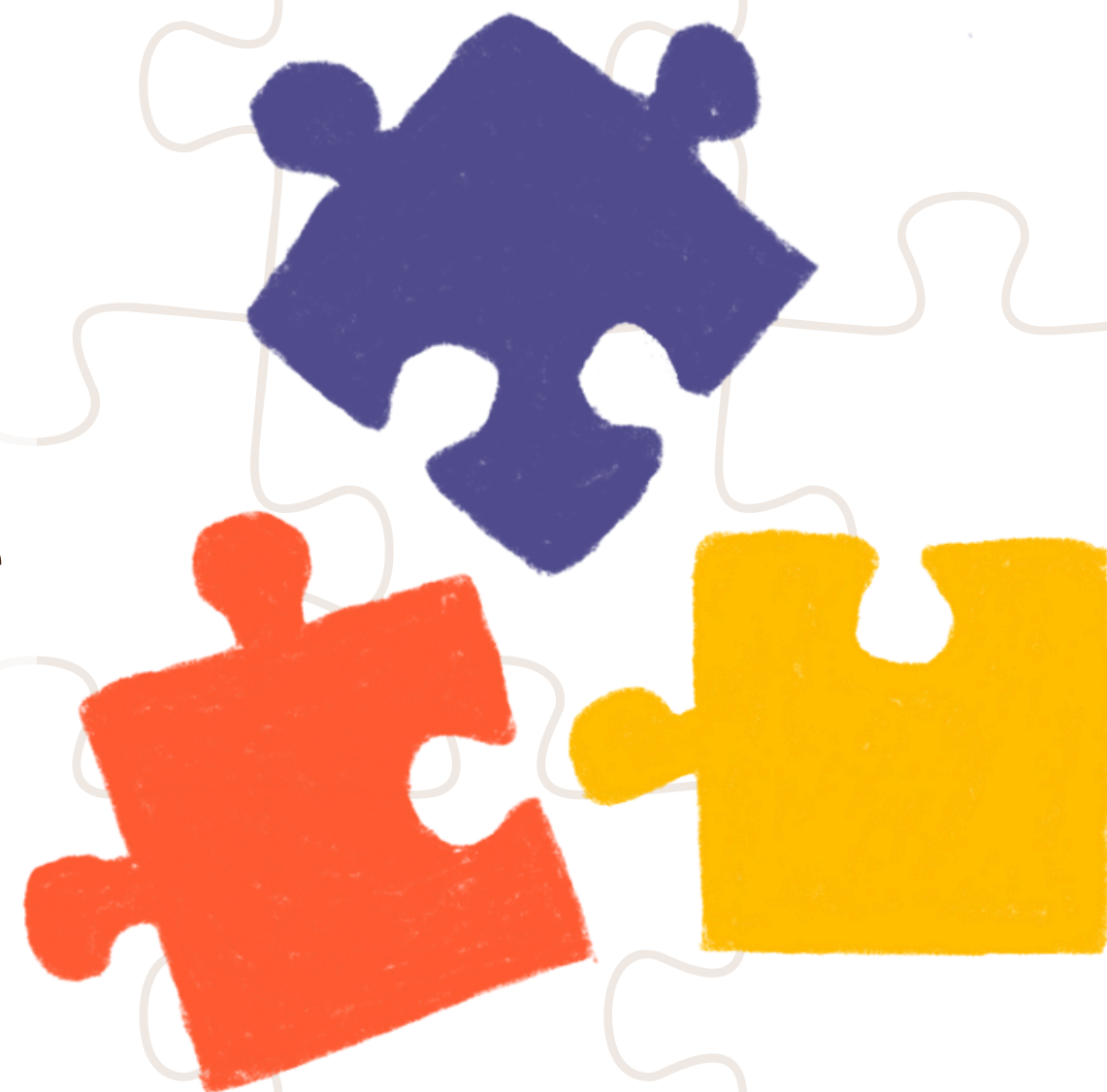


÷ DECOMPOSIÇÃO

Decomposição é o processo de quebrar um problema complexo em partes menores e mais gerenciáveis.

Digamos que você queira criar um aplicativo de lista de tarefas. Você pode decompor o problema em etapas menores, como adicionar uma tarefa, remover uma tarefa e marcar uma tarefa como concluída.

Imagine que você tem um grande quebra-cabeça para resolver, mas não sabe por onde começar. A decomposição é como se você pegasse esse quebra-cabeça enorme e o dividisse em pedaços menores e mais fáceis de resolver.



÷ RECONHECIMENTO DE PADRÕES

Reconhecimento de Padrões envolve identificar regularidades ou similaridades em dados, problemas ou situações.

É quando você vê algo se repetindo ou sendo parecido em diferentes situações. É como se seu cérebro dissesse: 'Ei, já vi isso antes!'.

Por exemplo, imagine que você está jogando um jogo de memória. Você vira as cartas e vê duas imagens iguais. Seu cérebro reconhece o padrão - as duas imagens são iguais - e você sabe onde encontrar o par correspondente.

Atente-se aos padrões - eles podem te ajudar a resolver o problema de forma mais rápida e eficiente!



÷ ABSTRAÇÃO

Abstração é o processo de ignorar detalhes irrelevantes e se concentrar nos aspectos essenciais de um problema.

Sabe quando você está assistindo a um filme ou jogando um jogo de tabuleiro e está tão envolvido na história ou na aventura que esquece do mundo ao seu redor? Isso é um pouco como a abstração!

Ao criar um jogo de vídeo, você abstrai os detalhes de baixo nível, como gerenciamento de memória, e se concentra na jogabilidade e na experiência do usuário.



÷ ALGORITMOS

Algoritmos são sequências de passos lógicos bem definidos para resolver um problema ou executar uma tarefa.

Imagine que você está seguindo as instruções para montar um brinquedo. As instruções te dizem exatamente o que fazer em cada etapa, desde abrir a caixa até ter o brinquedo pronto para brincar. Essas instruções são como um algoritmo!

Um algoritmo é como uma lista de passos que você segue para fazer algo. É como um mapa que te guia até o seu destino. São super importantes porque nos ajudam a automatizar processos e resolver problemas de forma eficiente.

```
state={
  products: storeProducts
}
render() {
  return (
    <React.Fragment>
      <div className="py-5">
        <div className="container">
          <Title name="our" title="product">
            <div className="row">
              <ProductConsumer>
                {(value) => {
                  console.log(value)
                }}
              </ProductConsumer>
            </div>
          </div>
        </div>
      </React.Fragment>
    )
  }
}
```


◆ Aplique o pensamento computacional na organização dos seus estudos.

Decomposição de Objetivos:

Quebre seus objetivos de aprendizado em partes menores. Por exemplo, se você deseja aprender uma nova linguagem de programação, divida o processo em etapas:

- 1 instalação do ambiente de desenvolvimento;
- 2 compreensão dos conceitos básicos;
- 3 prática de programação;
- 4 construção de projetos.

Reconhecimento de Padrões:

Identifique padrões em seu processo de aprendizado. Por exemplo, observe se você tem mais facilidade em aprender por meio de tutoriais em vídeo, documentação oficial ou livros. Reconhecer esses padrões pode ajudá-lo a otimizar sua abordagem de estudo.

Além disso, ao aprender diferentes conceitos de programação, observe os padrões que surgem. Por exemplo, identifique como loops são usados para repetir tarefas e como condicionais ajudam a tomar decisões no código.

◆ Aplique o pensamento computacional na organização dos seus estudos.

Abstração de Recursos:

Concentre-se nos conceitos fundamentais e essenciais da programação, como variáveis, funções e estruturas de dados, em vez de se perder em detalhes menos importantes. Isso ajudará a construir uma base sólida para sua compreensão.

Existem inúmeras ferramentas, livros, cursos, jogos e tutoriais disponíveis, mas nem todos serão úteis para você no momento. Abstraia os recursos que são mais adequados para o seu nível de habilidade e objetivos específicos.

Algoritmos de Estudo:

Desenvolva algoritmos eficientes para o seu estudo. Pense em uma sequência de passos lógicos para resolver o problema. Comece com uma abordagem simples e, em seguida, refine seu algoritmo à medida que ganha mais experiência.

Por exemplo, estabeleça um cronograma de estudo semanal que inclua tempo dedicado à revisão, prática de código, resolução de problemas e construção de projetos.

Adapte seu algoritmo de estudo conforme necessário com base no seu progresso e feedback.

◆ A prática é fundamental



Reserve um tempo regularmente para seus estudos. Isso ajudará a reforçar seus conhecimentos e a desenvolver habilidades práticas.


Encontre os materiais que se encaixam melhor com o seu jeito de aprender. Cada passo que você dá te leva mais perto dos seus objetivos. Continue aprendendo e evoluindo!

Não tenha medo de pedir ajuda e colaborar com outras pessoas. Participe de fóruns online, grupos de estudo ou eventos de programação para aprender mais e expandir sua rede de contatos na área.


O pensamento computacional é uma super-habilidade que pode ajudar todo mundo a resolver problemas de um jeito mais rápido.

Para saber mais:

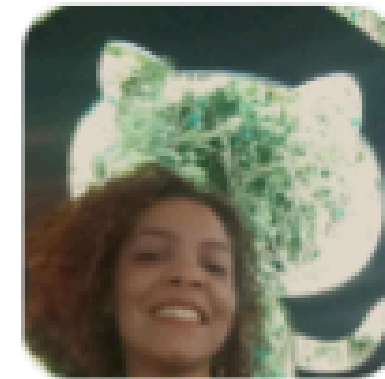
"Computational Thinking: A Beginner's Guide to Problem-Solving and Programming" por Karl Beecher.

Este livro oferece uma introdução acessível ao pensamento computacional, explorando como abordar problemas de forma sistemática usando conceitos como decomposição, reconhecimento de padrões, abstração e algoritmos. 

"Algorithm Design Manual" por Steven S. Skiena.

Este livro é uma referência valiosa para aqueles que desejam aprofundar sua compreensão de algoritmos e estruturas de dados, dois componentes fundamentais do pensamento computacional. Ele oferece uma abordagem prática e abrangente para projetar e analisar algoritmos. 

oliveiraas/**prompots- chatgpt-ebook**



Trabalho de conclusão de curso “IA for Devs”.
Realização: Dio Cursos Apoio: Banco Santander



1

Contributor



0

Issues



0

Stars



0

Forks



oliveiraas/prompots-chatgpt-ebook: Trabalho de conclusão de curso “IA for Devs”. Realização: Dio Curso...

Trabalho de conclusão de curso “IA for Devs”. Realização: Dio Cursos
Apoio: Banco Santander - oliveiraas/prompots-chatgpt-ebook



GitHub

