SUMÁRIO

Introdução à Programação e Pensamento Computacional

Curso base para iniciante, para quem quer começar a programar.

Itens que serão abordados:

- Pensamento Computacional;
- Introdução a Lógica da Programação;
- Fundamentos de Algoritmos;
- Linguagens de Programação;
- Primeiro Contato com a Programação.

Pensamento Computacional

O Pensamento Computacional (PC) é um processo de resolução de problemas que envolve a formulação de um problema e sua solução de forma que um computador possa executá-la. É um conjunto de habilidades que nos permite pensar como um computador para resolver problemas.

O PC pode ser usado para resolver uma ampla variedade de problemas, desde tarefas cotidianas até problemas científicos complexos.

Processo de pensamento está envolvida na expressão de soluções em passos computacionais ou algoritmos que podem ser implementados no computador.

O passo a passo de definir instrução para resolver um problema, as instruções irão definir a resolução dos problemas e essas instruções não se restringem aos computadores.

A solução de uma instrução deve ser resolvível por uma máquina e por um ser humano.

Pensamento computacional não é uma disciplina acadêmica e sim uma habilidade generalista que pode ser utilizada em todas áreas como por exemplo, matemática, leitura, escrita, etc..

O pensamento computacional está baseado em 4 pilares.

- Decomposição;
- Reconhecimento de padrões;
- Abstração;
- Design de algoritmos.

Decomposição

Dividir um problema em partes menores e mais gerenciáveis, a decomposição consiste em dividir um problema complexos em subproblemas, esta divisão proporciona facilidade e a melhora na resolução dos problemas em pontos menores e específicos.

Reconhecimento de padrões

Identificar padrões em dados e informações. Identificar similaridades e tendências dentro de um contexto ou de contextos distintos.

Padrão comportamental é um conjunto de regras e normas que definem como um sistema ou componente deve se comportar em determinadas situações. Ele serve como um guia para garantir que o sistema funcione de forma consistente e previsível.

Abstração

Identificar os aspectos essenciais de um problema e ignorar detalhes irrelevantes. Consiste em extrapolar um problema conceito de algum problema específico em uma forma generalista, ou seja, pego do mundo concreto e levo para o conceito das ideias.

Design de algoritmos

É a automatização da resolução dos problemas. Neste pilar é desenvolvido um conjunto de instruções passo a passo para resolver um problema. os algoritmos possuem suas etapas Entrada - Operações - Saída.

O processo de criação de algoritmos é contínuo e sempre precisa de refinamento, após a definição de uma solução esta solução deve ser testada e posteriormente deve ser aperfeiçoada gerando um ciclo virtuoso de refinamento, teste e análise.

Um algoritmo é uma sequência finita de instruções ou regras bem definidas e não ambíguas para resolver um problema ou realizar uma tarefa específica.

O pensamento computacional possibilita a utilização do melhor dos dois mundos entre as habilidades humanas e as habilidades recursos computacionais.

Os seres humanos são ótimos em identificar padrões e os computadores são melhores na resolução dos problemas.

Outras Competências

Outras competências adquiridas através do PC:

- Pensamentos Sistemáticos;
- Colaboração dentro da equipe;
- Criatividade e design;
- Facilitador.

Raciocínio Lógico

Raciocínio lógico é uma forma de pensamento estruturado, ou raciocínio, que permite encontrar a conclusão ou determinar a resolução de um problema.

O raciocínio lógico é a capacidade de pensar de forma clara, organizada e precisa para chegar a conclusões válidas. É como uma ferramenta que nos permite analisar informações, identificar falhas e tomar decisões sensatas.

O raciocínio lógico deve ser treinado trata-se de uma habilidade que com o treinamento ele fica mais intuitivo e internalizado.

O raciocínio lógico está classificado em 3 grupos, indução, dedução e abdução.

Indução

Vem a partir de um fenômeno observado, e a partir do fenômeno observado você consegue extrapolar e determinar leis e teorias relacionados ao fenômeno. Este tipo de técnica está relacionado a ciências experimentais.

Parte de observações específicas para chegar a conclusões gerais. É como subir uma escada, onde cada degrau representa uma nova observação que leva a uma conclusão mais ampla.

Exemplo:

Observação 1: O cisne que vi é branco.
Observação 2: Outro cisne que vi é branco.
Conclusão: Todos os cisnes são brancos.

Indutivo em resumo: Formular hipóteses e fazer previsões.

Dedução

A partir de leis e teorias é deduzido previsões e explicações para os fenômenos, é o contrário da indução. Este tipo de raciocínio lógico é utilizado por exemplo nas ciências exatas.

Começa com premissas gerais e utiliza regras lógicas para chegar a conclusões específicas. É como um funil, onde as premissas abrangem um universo maior e a conclusão é um caso particular dentro desse universo.

Exemplo:

Premissa 1: Todos os humanos são mortais.
Premissa 2: Sócrates é humano.
Conclusão: Sócrates é mortal.

Dedutivo em resumo: Provar ou refutar uma afirmação.

Abdução

Neste grupo de raciocínio lógico, a partir de uma conclusão você extrai uma premissa.

Propõe hipóteses para explicar fatos observados. É como um detetive juntando pistas para formular a melhor explicação para um crime.

Exemplo:

Fato: A grama está molhada.
Hipótese 1: Choveu.
Hipótese 2: Alguém jogou água na grama.

Este tipo de técnica é utilizada em processo investigativo de diagnósticos.

Abdutivo em resumo: Explicar fatos observados.

Inferência

A inferência é um aspecto do raciocínio lógico, a inferência é o processo de chegar a uma conclusão a partir de informações disponíveis. É como usar pistas para desvendar um mistério ou completar um quebra-cabeça.

Refere-se ao processo mental de deduzir ou concluir algo com base em evidências, observações ou informações disponíveis. É a capacidade de chegar a uma conclusão lógica com base em premissas ou fatos conhecidos. A inferência pode ocorrer de várias maneiras, incluindo dedução, indução e abdução, como mencionado anteriormente. É mais específico, relacionando-se diretamente à extração de conclusões a partir de informações disponíveis.

Na classificação tradicional de Kant, a inferência é dividida em dois tipos principais: inferência sintética e inferência analítica.

Inferência Sintética

Na inferência sintética, a conclusão vai além das premissas fornecidas e adiciona novas informações. É uma expansão do conhecimento além do que já é conhecido.

Relaciona-se principalmente com o raciocínio indutivo e abdutivo, que parte de observações específicas para chegar a conclusões gerais. A inferência sintética muitas vezes envolve a generalização a partir de casos particulares para chegar a conclusões mais amplas.

Exemplo: Se observarmos que todas as maçãs que vimos são vermelhas, podemos inferir sinteticamente que todas as maçãs são vermelhas.

Inferência Analítica

Na inferência analítica, a conclusão está contida nas premissas fornecidas. Ela se limita a explicar o que já está implícito nas premissas, sem adicionar novas informações.

Relaciona-se principalmente com o raciocínio dedutivo, que parte de premissas gerais para chegar a conclusões específicas. A inferência analítica envolve a dedução de conclusões necessárias a partir de premissas dadas.

Exemplo: Se todas as maçãs são frutas e algo é uma maçã, podemos analiticamente inferir que é uma fruta.

Exemplo de Raciocínio Lógico

Imagine que você está com Sherlock Holmes investigando um crime. A vítima, um famoso pianista, foi encontrado morto em seu apartamento. As pistas são intrigantes: uma partitura rabiscada, um vaso quebrado e uma testemunha que viu um vulto saindo do local do crime.

Usando o raciocínio lógico, podemos desvendar o mistério:

1. Observação e Análise:

Observamos cuidadosamente as pistas: a partitura, o vaso e o depoimento da testemunha.

Analisamos cada pista em busca de detalhes relevantes:

- A partitura tem rabiscos que podem ser um código secreto.
- O vaso quebrado pode ter sido usado como arma.
- A testemunha não conseguiu identificar o vulto.

2. Formulação de Hipóteses:

Com base nas pistas, formulamos hipóteses sobre o que pode ter acontecido:

- O pianista foi morto por um rival que invejava seu talento.
- Um fã obcecado invadiu o apartamento e o matou.
- O crime foi resultado de um assalto que deu errado.

3. Teste das Hipóteses:

Para cada hipótese, buscamos evidências que a confirmem ou refutam:

- Investigamos o passado do pianista em busca de rivais.
- Entrevistamos fãs e amigos do pianista para verificar se alguém demonstrava obsessão.
- Procuramos por impressões digitais no vaso quebrado e na partitura.

4. Dedução e Conclusões:

A partir da análise das evidências, podemos deduzir qual das hipóteses é mais provável:

- Se encontrarmos provas de um rival com rancor, a primeira hipótese se torna mais forte.
- Se a testemunha reconhecer o vulto como um fã conhecido, a segunda hipótese ganha força.
- Se as impressões digitais no vaso e na partitura forem de um ladrão conhecido, a terceira hipótese se torna a mais plausível.

5. Raciocínio Lógico em Ação:

Neste exemplo, utilizamos os seguintes tipos de raciocínio lógico:

- Dedutivo: Para deduzir qual a hipótese mais provável com base nas evidências.
- Abdutivo: Para formular hipóteses plausíveis que expliquem as pistas.
- Indutivo: Para generalizar a partir das pistas e formular uma conclusão sobre o crime.

O raciocínio lógico é uma ferramenta poderosa que nos permite desvendar mistérios, tomar decisões e resolver problemas. Através da observação, análise, formulação de hipóteses, teste e dedução, podemos chegar a conclusões válidas e tomar medidas eficazes.

Aperfeiçoamentos

A partir de uma solução, determinar pontos de melhora e refinamento sejam eles pontuais ou de uma maneira global.

Dentro do ato de aperfeiçoar temos que encontrar soluções eficientes e a otimização de processo (melhor uso de recurso) ou aperfeiçoamentos de simplificação de linhas de códigos e funções bem definidas (melhorar códigos e algoritmos).

Pilares: Decomposição

A decomposição é um dos pilares fundamentais do pensamento computacional, uma habilidade essencial para solucionar problemas complexos de forma eficiente.

Decomposição é o processo de dividir um problema grande e complexo em partes menores e mais gerenciáveis. Imagine uma enorme torre de blocos: a decomposição seria dividi-la em camadas, fileiras e blocos individuais.

Benefícios da Decomposição

- Simplificação: Torna o problema mais fácil de entender e resolver.
- Organização: Permite focar em uma parte de cada vez, evitando sobrecarga.
- Eficiência: Facilita a identificação de soluções e a implementação de cada etapa.
- Reutilização: As soluções para as partes menores podem ser reutilizadas em outros problemas.

Estratégias de Decomposição

Para realizar a decomposição de forma eficaz, podemos utilizar duas estratégias principais: análise e síntese.

Análise

Processo de quebrar e determinar partes menores e gerenciáveis.

- Desconstrução: Dividir o problema em seus componentes básicos, como etapas, funções, módulos ou classes.
- Identificação: Reconhecer os elementos inter-relacionados dentro de cada componente.
- Abstração: Ignorar detalhes irrelevantes e focar nas características essenciais de cada componente.

Estratégias de análise

- Top-down: Começar com a visão geral do problema e dividir em partes cada vez menores.
- Bottom-up: Começar com as partes menores do problema e combiná-las para formar a solução completa.
- Decomposição por função: Dividir o problema em partes com base em suas funções ou responsabilidades.
- Decomposição por dados: Dividir o problema em partes com base nos dados que cada parte utiliza.

Síntese

Combinar os elementos recompondo o problema original

- Combinação: Reunir as partes menores do problema em uma solução completa e coesa.
- Organização: Estabelecer a relação entre as partes menores e definir a ordem de execução.
- Validação: Testar a solução para garantir que ela resolva o problema original.

Estratégias de síntese

- Refinamento: Ajustar e aprimorar as partes menores para que se integrem perfeitamente.
- Iteração: Testar e corrigir a solução de forma incremental, ajustando as partes conforme necessário.
- Abstração: Criar representações abstratas das partes menores para facilitar a integração.

Ordem de execução da Decomposição

A ordem de execução do processo de decomposição pode ser sequencial ou paralela, dependendo da natureza do problema e dos recursos disponíveis.

Sequencial

No processo de execução sequencial existe dependência entre as tarefas.

- As partes do problema são executadas uma após a outra, em uma ordem específica.
- É ideal para problemas que possuem dependências entre as partes, ou seja, quando uma parte precisa ser concluída antes que a outra possa ser iniciada.

Exemplo: Construção de uma casa: as fundações precisam ser concluídas antes de iniciar a construção das paredes.

Paralelo

A aplicação do paralelismo ganha-se em eficiência e tempo de resolução do problema, pois a resolução das tarefas ocorrem de forma isoladas e independentes e após resolvidas são agregadas para resolver o problema maior.

- As partes do problema são executadas ao mesmo tempo, em diferentes threads ou processos.
- É ideal para problemas que possuem partes independentes que não dependem umas das outras.

Exemplo: Cálculo de impostos para vários produtos: os impostos para cada produto podem ser calculados em paralelo.

Dentro da decomposição temos as variáveis que estão dentro dos problemas pequenos que por sua vez fazem parte da segmentação do problema maior.

A decomposição deve ser desenvolvida pela própria pessoa para ganhar a habilidade de decompor os problemas. A decomposição pode ser realizada de formas diferentes para determinados problemas.

Como Decompor?

1. Identificar ou coletar os dados relacionados ao problema;
2. Agregar os dados;
3. Resolução e entrega da funcionalidade.

Exemplo de Decomposição

Definição das etapas para criação de um App, decompor em tópicos e definir e descrever qual a tarefa a ser executada em cada etapa.

- Finalidade
- Interface
- Funcionalidades
- Pré-Requisitos

Pilares: Padrões

O reconhecimento de padrões é um dos pilares fundamentais do pensamento computacional, permitindo identificar regularidades e similaridades em diferentes situações. Essa habilidade é essencial para solucionar problemas de forma eficiente, criativa e generalizável.

Um padrão é uma sequência, estrutura ou característica que se repete em diferentes contextos. Pode ser algo visual, como a forma de uma flor, ou algo abstrato, como o comportamento de um animal.

Um padrão possui:

- Modelo Base
- Estrutura Invariante
- Repetição

Com reconhecimento de padrões é possível a detecção de similaridade e referências.

Porque Determinar Padrões

No processo de determinação de padrões, a generalização é a etapa crucial que permite extrair princípios e regras universais a partir de casos específicos. É a ponte que conecta o concreto ao abstrato, abrindo caminho para a aplicação do conhecimento em diferentes contextos.

Generalizar significa identificar características comuns em diferentes situações e formular uma regra ou princípio que as abrange todas. É abstrair os detalhes irrelevantes e focar na essência do que está sendo observado.

Importância da Generalização

- Ampliação do Conhecimento: Permite aplicar o conhecimento adquirido em um contexto para outros contextos com características similares, expandindo o alcance do nosso aprendizado.
- Eficiência Imbatível: Facilita a resolução de problemas, pois não é necessário reinventar a roda a cada novo desafio.

- Compreensão Profunda: Promove uma compreensão mais profunda dos princípios subjacentes aos padrões, permitindo uma aplicação mais flexível e adaptável.

A resolução de problemas através do desenvolvimento de uma funcionalidade para resolução de um determinado problema com um padrão reconhecido pode ser replicado a outros casos.

Reconhecimento de Padrões - Seres Humanos

A tarefa de reconhecimento de padrão pelos seres humanos é intuitiva através de similaridade entre os objetos.

Os seres humanos fazem o reconhecimento por:

- grau de similaridade
- grupos conhecidos x objeto desconhecido

O reconhecimento de padrões em humanos é um processo complexo que envolve diversas áreas do cérebro e diferentes etapas:

1. Percepção

- Recepção de informações: Através dos sentidos, como visão, audição e tato, coletamos informações do ambiente.
- Processamento sensorial: O cérebro processa essas informações, reconhecendo formas, cores, texturas e outros elementos básicos.

2. Memória

- Armazenamento de padrões: A memória armazena padrões já conhecidos, como o rosto de um amigo ou o som de um carro.
- Comparação com novos padrões: Ao encontrar um novo padrão, o cérebro o compara com os padrões armazenados na memória.

3. Abstração

- Identificação de características essenciais: O cérebro identifica as características essenciais do novo padrão, ignorando detalhes irrelevantes.
- Categorização: O novo padrão é categorizado em uma classe já existente ou em uma nova classe.

4. Tomada de decisão

- Utilização de padrões: Com base nos padrões reconhecidos, o cérebro toma decisões sobre como agir ou interpretar a situação.

Reconhecimento de Padrões - Computadores

O reconhecimento de padrões é um campo fundamental que permite que os computadores identifiquem regularidades e similaridades em dados, imagens, sons e outros tipos de informações. Essa habilidade é essencial para que as máquinas aprendam a realizar tarefas complexas como:

- Visão computacional: Identificar objetos, pessoas e faces em imagens e vídeos.

- Processamento de linguagem natural: Compreender o significado de textos e frases.
- Aprendizagem de máquina: Extrair conhecimento de grandes conjuntos de dados.
- Robótica: Controlar o movimento de robôs e interagir com o ambiente.

Como o computador reconhece os padrões?

Os computadores reconhecem padrões através de um processo fascinante que combina matemática, algoritmos e inteligência artificial. Essa habilidade permite que as máquinas identifiquem regularidades e similaridades em dados, imagens, sons e outros tipos de informações, abrindo um mundo de possibilidades para a resolução de problemas complexos.

Como simular esse comportamento de reconhecimento de Padrões?

- Representação de atributos
- Aprendizado - Conceito Associado ao objeto
- Armazenar Dados
- Regras de Decisão

Aplicações que utilizam os reconhecimento de padrões

- Classificação de dados
- Reconhecimento de imagem
- Reconhecimento de fala
- Análise de cenas
- Classificação de documentos
- Biometria
- Análise de Dados

Áreas que utilizam o reconhecimento de padrões

- Machine Learning
- Redes Neurais
- Inteligencia Artificial
- Ciência de Dados

Pilares: Abstração

Abstração em pensamento computacional refere-se à capacidade de identificar e enfatizar os aspectos importantes de um problema ou situação, enquanto suprime detalhes irrelevantes ou menos importantes. É um conceito fundamental na ciência da computação e no desenvolvimento de algoritmos.

Na prática, a abstração envolve criar modelos simplificados que representam sistemas complexos ou problemas do mundo real. Esses modelos permitem que os computadores resolvam problemas de forma eficiente, ao mesmo tempo que tornam mais fácil para os humanos entenderem e lidarem com a complexidade.

Abstrair

É a ação de separar mentalmente algo de sua concretude ou de suas características específicas, focalizando apenas nos aspectos essenciais ou conceituais. Quando alguém abstrai, está simplificando um conceito,

removendo detalhes desnecessários para entender o seu cerne ou essência. É o ato de observar um ou mais elementos avaliando as características e propriedades em separado.

Generalizar

É a ação de extrair ou formular uma ideia, conceito ou padrão que se aplica a uma variedade de situações, objetos ou eventos. Quando alguém generaliza, está buscando identificar características comuns em diferentes casos específicos e formular princípios ou regras gerais que se aplicam a todos esses casos, tornando algo geral, mais amplo, extenso.

Na lógica a generalização é a operação que consiste, por exemplo, em reunir numa classe geral um conjunto de seres ou fenômenos similares.

Na lógica de programação, o processo de generalização envolve identificar padrões ou características comuns em um conjunto de dados ou situações específicas e, em seguida, formular regras ou algoritmos que se apliquem a uma variedade de casos. Em outras palavras, é a capacidade de abstrair os elementos essenciais de um problema para criar soluções mais amplas e flexíveis.

Na lógica de programação, a generalização é o processo de criar código que pode ser aplicado a diferentes situações e conjuntos de dados. Em outras palavras, é como construir uma caixa de ferramentas versátil que pode ser usada para resolver diversos problemas.

Como classificar os dados a partir da abstração

Classificar dados a partir da abstração envolve identificar e agrupar elementos semelhantes ou relacionados com base em características compartilhadas. Esse processo pode variar dependendo do tipo de dados e do contexto em que estão sendo trabalhados.

Ela envolve identificar características comuns e padrões nos dados e agrupá-los em categorias hierárquicas com base em diferentes níveis de abstração.

- Identificar as características relevantes: Antes de começar a classificação, é importante entender quais características dos dados são significativas para o agrupamento. Isso pode incluir atributos como cor, tamanho, tipo, valor, categoria, etc., dependendo do tipo de dados.
- Pontos Essenciais: Concentre-se nos pontos essenciais que distinguem os diferentes grupos de dados. Por exemplo, ao classificar frutas, podemos generalizar que frutas com pele vermelha são diferentes das frutas com pele amarela.
- Generalizar em detrimento do detalhe: Ao classificar os dados, é importante generalizar, ou seja, enfatizar as semelhanças em detrimento aos detalhes específicos.

Representação de dados a partir da abstração

A representação de dados a partir da abstração envolve simplificar a complexidade dos dados, destacando apenas os aspectos essenciais que são relevantes para o problema em questão. Identificar os pontos essenciais e descartar os detalhes.

Conceitos Baseados em Abstração

- Modelagem de Dados: Na modelagem de dados, a abstração é usada para representar entidades, atributos e relacionamentos de uma forma que simplifique a compreensão e manipulação dos dados. Modelos conceituais, como diagramas de entidade-relacionamento, fornecem uma representação abstrata dos dados que ignoram os detalhes de implementação.

- Estruturas de Dados Abstratas: Estruturas de dados abstratas, como listas, conjuntos, filas e árvores, são conceitos baseados em abstrações que representam maneiras de organizar e acessar dados de forma eficiente. Elas ocultam os detalhes de implementação subjacentes e fornecem uma interface consistente para manipulação de dados.

- Programação Orientada a Objetos: Na programação orientada a objetos, os conceitos de classe e objeto são baseados em abstração. Uma classe representa um tipo de objeto e define seus atributos e comportamentos, enquanto um objeto é uma instância específica dessa classe. A abstração permite encapsular dados e funcionalidades relacionadas em unidades independentes e reutilizáveis.

- Algoritmos e Estruturas de Controle: Algoritmos e estruturas de controle, como loops e condicionais, são conceitos baseados em abstrações que permitem expressar operações complexas em termos mais simples. Eles abstraem os detalhes de implementação dos passos individuais necessários para resolver um problema e fornecem uma maneira de expressar o fluxo de execução de forma clara e concisa.

- Padrões de Projeto: Padrões de projeto são soluções reutilizáveis para problemas comuns de design de software. Eles são construídos sobre o princípio da abstração, encapsulando os detalhes de implementação específicos em uma interface genérica que pode ser aplicada em diferentes contextos. Os padrões de projeto ajudam a promover a reutilização de código, modularidade e flexibilidade do sistema.

Pilares: Algoritmos

Um algoritmo é uma sequência finita e bem definida de instruções que resolve um problema específico. É como uma receita culinária que descreve passo a passo como preparar um prato, ou um manual que explica como montar um móvel.

Em outras palavras:

- Um conjunto de instruções que um computador pode seguir para realizar uma tarefa.
- Uma ferramenta poderosa para resolver problemas de forma eficiente e automatizada.
- Uma linguagem precisa e universal para descrever processos e soluções.

Um algoritmo não opera sozinho, deve ser determinar instruções detalhadas para que ele funcione.

- sequência de passos com objetivos definido
- execução de tarefas específicas
- conjunto de operações que resultam em uma sucessão finita de ações

Processamento de Dados

O computador recebe, manipula e armazena dados através dos programas. Os programas por sua vez são constituídos pelas instruções que possuem o passo a passo de como executar uma determinada tarefa.

O processo de resolução de problemas ocorre “step by step” utilizando instruções determinadas.

O algoritmo deve ser entendido por um humano e pela máquina.

Desenvolvimento do Programa

O desenvolvimento de um programa geralmente segue várias etapas, desde a concepção da ideia até a implementação e manutenção do software.

- Analise
    - Identificação da necessidade ou problema a ser resolvido pelo software.
    - Análise dos requisitos do sistema e definição das funcionalidades necessárias para atender a esses requisitos.
    - Desenvolvimento de uma especificação de software que descreve detalhadamente o que o programa deve fazer.
- Algoritmo
    - Descreve o problema por meio de ferramentas narrativas, fluxogramas e pseudocódigo
- Codificação
    - O algoritmo é codificado com a linguagem de programação escolhida.

Exemplos de Algoritmos

Organizar uma lista de nomes em ordem alfabética:

- Comece com o primeiro nome da lista.
- Compare-o com o segundo nome.
- Se o primeiro nome for menor, troque os dois nomes de lugar.
- Repita os passos 2 e 3 até que todos os nomes estejam em ordem.

Como construir um Algoritmos

Construir um algoritmo envolve seguir uma sequência lógica de passos para resolver um problema específico.

- Compreensão do problema
- Definição dos dados de entrada
- Definir processamento
- Definir dados de saída
- Utilizar um método de construção
- Testes e diagnósticos

A construção de um algoritmos ainda envolve os passos abaixo:

- Narrativa - Utilização de linguagem natural para definição e descrição do que o algoritmo deve executar.
- Fluxograma - Representação gráfica, símbolos e setas visuais para ilustrar a sequência de passos do algoritmo.
- Pseudocódigo - É uma linguagem descritiva com instruções, passo a passo em linguagem natural simplificada. É o mais próximo da linguagem de programação.

Estudo de Caso Conceitual - Perdido

Como resolver o problema utilizando o pensamento computacional

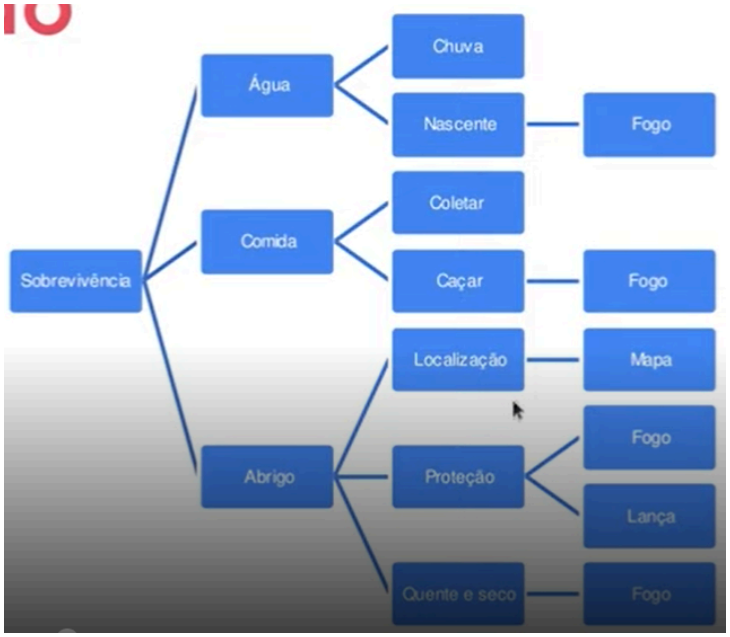
- Identificar mecanismos
- recursos comuns
- detalhes mais importantes

Se estiver perdido na floresta quais as necessidades para sobrevivência?

Água / comida / Abrigo

O problema da sobrevivência foi decomposto em problemas menores (Água / comida / Abrigo).

Decompondo ainda mais conseguimos maiores detalhes em partes ainda menores menores.



Na análise desta decomposição verificamos que o fogo é algo importante pois se repete em vários níveis.

O foco tem que ser focado nos aspectos principais, não precisa de detalhamento.

Neste detalhe conseguimos decompor para segmentar o problema.

O próximo passo seria detalhar as instruções passo a passo para por exemplo realizar a comida.

Neste exemplo foi utilizado todos os pilares de pensamento computacional decomposição, encontro de padrões, abstração e algoritmos.

Estudo de Caso Aplicado - Soma de um intervalo

Estudo de caso soma de intervalo.

Some de n° entre 1 e 200.

Uma solução seria ir somando sequencialmente os números 1+2, 1+3, 1+4... Porém esta solução seria ineficiente.

Outra forma seria a soma do menor com o maior e ir decrementando o maior e incrementando o menor nas somas 200+1, 199+2, 198+3 ...

Com esta última forma temos um padrão, toda vez que é somado o maior valor e o menor valor a soma sempre é 201.

Com a decomposição do problema (200+1, 199+2, 198+3...) e o padrão (resultado da soma sempre 201) podemos expressar de forma generalista e abstrair para que possamos chegar no resultado de forma ainda mais eficiente.

Na soma entre os n° de 1 e 200 na decomposição (200+1, 199+2, 198+3...) por estar utilizando 2 números a cada soma então eu divido os 200 números por 2 (200 / 2 = ***100***)

Por tanto o resultado da soma entre os n° de 1 e 200 seria 201 x 100 = 20.100.

Devo expressar esta soma de intervalos específicos em variáveis, para algo generalista para utilizar este cenário para outras somas.

Exemplo soma de n° entre x e y

[x, Y] - Intervalo soma
y + x = resultado parcial
y / 2 = total
total * resultado parcial = resultado da soma entre os intervalos

Agora após a decomposição, reconhecimento de padrões e a abstração podemos criar o algoritmo:

Passo 1 - Receber os valores (x e y)
Passo 2 - Resolva:
    y / 2 = total
Passo 3 - Resolva:
    y + x = resultado_parcial
Passo 4 - Ache o total
    Final = total x resultado_parcial
Passo 5 - Imprima o resultado

Estudo de Caso Aplicado - Adivinhe o Número

Determine o número escolhido por uma pessoa dentro de um intervalo, com perguntas com respostas de sim e não.

Possível Solução

Maneira ineficiente e com desperdício de tempo, perguntar número a número se o número foi escolhido.

Maneira eficiente seria perguntas se o número é maior que 50 e sendo a resposta não podemos então questionar se o número é maior que 20 e assim sucessivamente até adivinhar o número.

Podemos utilizar a Busca binária.

Para aprimorar a habilidade de raciocínio lógico deve permitir discussões e explicação sobre como foi pensado o pensamento computacional.

Introdução a Lógica da Programação

Fundamentos de Algoritmos

Linguagens de Programação

Primeiro Contato com a Programação