BOOTCAMP SANTANDER FULLSTACK

“Processo de pensamento envolvido na expressão de soluções em passos computacionais ou algoritmos que podem ser implementados no computador.”

**Pensamento Computacional**

* Sistemático e eficiente
* Formulação e resolução de problemas
* Sejam capazes de resolver
* Humanos & máquinas

Baseado em 4 pilares:

1. **Decomposição** – consiste em dividir um problema complexo em subproblemas
2. **Reconhecimento de padrões** – identificar padrões ou tendências entre os problemas
3. **Abstração** – extrapolar o conceito do problema para uma forma generalista
4. **Design de algoritmos** – **automatizar**: definir passo a passo a solução do problema

**Habilidades Complementares**

* Raciocínio Lógico
* Aperfeiçoamento

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

1. Indução – leis e teorias através da observação. Ciências experimentais
2. Dedução – leis e teorias através de previsões e explicações. Ciências exatas
3. Abdução – conclusão através da premissa do problema. Processo investigativo

ANALÍTICA

ABDUÇÃO

INDUÇÃO

SINTÉTICA

DEDUÇÃO

INFERÊNCIA

**Aperfeiçoamento**

* Encontrar solução eficiente

Melhor uso de recursos

* Otimizar processos
* Simplificar linhas de códigos

Melhorar códigos e algoritmos

* Funções bem definidas

**Pilar - Decomposição**

Primeiro passo da resolução de problemas dentro do conceito de pensamento computacional

**Estratégia**

Processo de quebrar e determinar partes menores e gerenciáveis - Análise

Combinar os elementos recompondo o problema original - Síntese

Ordem de execução de tarefas menores:

1. Sequencial: dependência entre tarefas
2. Paralelo: tarefas podem ser executadas ao mesmo tempo

**Como decompor?**

1. Identificar ou coletar dados
2. Agregar os dados
3. Funcionalidade

Ex: cozinhar. Identificar os ingredientes > determinar as etapas > executar cada etapa > agregar os ingredientes para finalizar

Ex: criar um app. Finalidade > interface > funcionalidades > pré-requisitos = definição de componentes e etapas

**Pilar - Padrões**

Reconhecimento de padrões:

* Modelo base
* Estrutura invariante
* Repetição

**Finalidade**: generalizar, com objetivo de obter resolução para problemas diferentes

O computador **reconhece os padrões** através da **comparação**.

1. Extração de características
2. Classificação de dados

Áreas que utilizam padrões em suas aplicações

* Machine Learning
* Redes Neurais
* Inteligência Artificial
* Ciência de dados

**Pilar – Abstração**

Processo intelectual de isolamento de um objeto da realidade

Generalizar: torna algo geral, mais amplo e extenso. **Abstrair é generalizar**

Generalizar: consiste em reunir numa classe geral um conjunto de seres ou fenômenos similares

**Classificação dos dados:**

* Características
* Pontos essenciais
* Generalizar x detalhar

**Pilares – Algoritmos**

Não opera sozinho.

Precisa de instruções detalhadas.

O que precisa ser feito?

instruções

Qual a ordem de execução?

**Desenvolvimento do Programa**

* Análise – estudo e definição dos dados de entrada e saída
* Algoritmo – descreve o problema por meio de ferramentas narrativas, fluxograma ou pseudocódigo
* Codificação – o algoritmo é codificado de acordo com a linguagem de programação escolhida

**Sequência de passos com objetivo definido**

**Execução de tarefas específicas**

**Conjunto de operações que resultam em uma sucessão finita de ações**

Ex: para preparar um sanduíche, é necessário seguir um passo a passo, assim como para criar um programa.

**Como construir um algoritmo?**

1. Compreensão do problema
2. Definição dos dados de entrada (fornecidos e cenário)
3. Definir processamento (cálculos e restrições)
4. Definir dados de saída (após processamento)
5. Utilizar um método de construção (construção e refinamento do algoritmo)
6. Teste e diagnóstico

**Construção de algoritmos**

**Narrativa** – utiliza a linguagem natural – diversas interpretações possíveis

**Fluxograma** – utilização de símbolos pré-definidos – conhecimento prévio da estrutura e símbolos

**Pseudocódigo** – “portugol” – passos a serem seguidos

**Estudo de caso – soma de intervalo**

Ex: soma de número entre 1 e 200

1+2

1+3

1+4... **Solução ineficiente**

200+1

199+2

198+3... **Decomposição**

200+1 = 201

199+2 = 201

198+3 = 201 **Padrão**

**O valor 201 vai se repetir. Quantas vezes? 200/2 = 100**

**Resultado: 201 x 100 = 20.100**

Ex: soma de número entre x e y

[x, y] – intervalo **1 e 200**

Y+x = resultado parcial

(y-1) + (x+1) = resultado parcial **200+1 = 201 199+2 = 201**

**y/2 = total (200/2 = 100)**

**total x resultado parcial = resultado**

**resultado: 100x201 = 20.100**

1. Recebe os valores (x e y)
2. Resolva: y/2 = total
3. Resolva: y+x = resultado\_parcial
4. Ache o total. Final = total x resultado\_parcial
5. Imprima o resultado

**Estudo de caso aplicado: adivinhe o número**

Ex: adivinhe o número

O problema consiste em determinar o número escolhido por uma pessoa dentro de um intervalo

Perguntas com respostas de sim e não

Ex: **adivinhe o número**

O número é 1?

O número é 2?

O número é 3?... **Maneira ineficiente e demanda muito tempo**

O número é maior que 50? **Não**

O número é menor que 20? **Sim**

**Até encontrar o número**

**Ex: busca binária** – mais eficiente que busca por varredura. Método de busca

1. Ordenar o vetor
2. Módulo de L/2
3. Acessar estrutura
4. Comparar valores
5. Repita até encontrar o número
6. Imprima “Busca bem sucedida”