# Introdução à Programação e Pensamento Computacional

## 1. Pensamento Computacional

Baseado em quatro pilares: decomposição, reconhecimento de padrões, abstração e design de algotritmos.

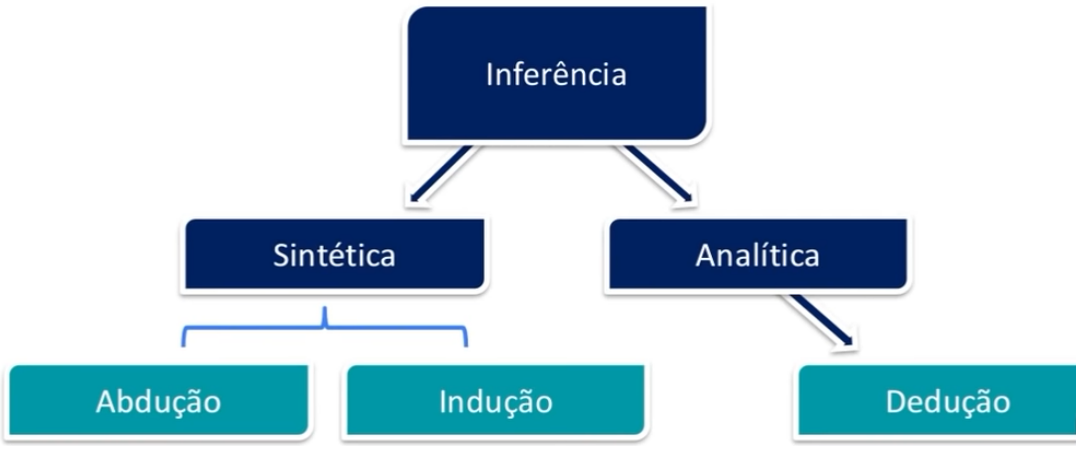
### 1.1 - Habilidades Complementares

#### Raciocínio Lógico (Habilidade de Treinamento)

Forma de pensamento estruturado, ou raciocínio, que permite encontrar a conclusão ou determinar a resolução de um problema.

Classificado em três grupos:

- Indução: a partir de um fenômeno observado é possível criar leis e teorias.
- Dedução: a partir de leis e teorias deduzimos o acontecimento de algo; prova de uma teoria.
- Abdução: a partir de algo observado, supomos outra coisa. Ex.:A grama está molhada, logo deve ter chovido. Diagnóstico, processo investigativo. Usado em casos de crimes, por exemplo.




#### Aperfeiçoamento

Ato de aperfeiçoar

- Encontrar solução eficiente
- Otimizar processos
- Simplificar linhas de códigos
- Funções bem definidas

## 1.2 Pilares - Decomposição

Primeiro passo para a resolução de um problema, dentro do pensamento computacional.

**Estrátégias**

Análise: processo de quebrar e determinar partes menores e gerenciáveis.
Síntese: combinar os elementos recompondo o problema original, de maneira coerente e dando sentido à sua solução.
Ordem de execução das tarefas: sequencial ou paralela, dependendo do contexto dos probelmas.

**Como decompor:**

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

## 1.3 Pilares - Padrões

Reconhecimento de padrões

- Modelo base
- Estrutura invariante
- Repetição

Determinar similaridades e diferenças entre os contextos e objetos

Os serer humanos determinam padrões através de similaridade e grupos conhecidos x desconhecidos.
Já o computador reconhece padrões através de comparação. Para isso devemos representar os atributos,  aprendizado pelo conceito associado ao objeto, para que ele armazene os dados e possa decidir.

## 1.4 Pilares - Abstração

"Processo intelectual de isolamento de um objeto da realidade." - extrapolar um objeto do mundo concreto para o mundo das ideias.

Abstrair = "Observar um ou mais elementos, avaliando características e propriedades em separado." - detectar
características.

Generalizar= Tornar mais amplo

Como classificar dados?

- Através de características
- Detectar os pontos essenciais
- Generalizar em detrimento do detalhe

### 1.5 Pilares - Algoritmos

O que precisa ser feito?
Qual a ordem da execução?

**Desenvolvimento do programa**

- Aálise: Quais são os dados de entrada e de saída?
- Algoritmo: desecreve o problema por meio de ferramentas narrativas, fluxograma ou pseudocódigo.
- Codificação

**Como construir um algoritmo**

- Compreensão do problema
- Definição de dados de entrada
- Definir processamento
- Definir dados de saída
- Utilizar um método de construção
- Teste e diagnóstico

**Narrativa -** Utilização da linguagem natural, sem conceitos novos; diversas interpretações possíveis.
**Fluxograma -** Utilização de símbolos pré definidos que definem o tipo de operação que está ocorrendo; simples entendimento; requer conhecimento prévio da estrutura e dos símbolos.
**Pseudocódigo -** Portugol; regras definidas, passo a passo a ser seguido.

# 2.Introdução à Lógica de Programação

### Técnicas de Lógica de Programação

**1. Técnica Linear**

- Modelo linear

- Não tem vínculo
  - Estrutura hierárquica
  - Programação de computadores
- Modelos de desnevolvimento e resolução
- Execução senquenciada de uma série de operações; ordenação  de elementos por uma única propriedade.
- Recusos limitados; única dimensão.

**2. Técnica Estruturada**

- Organização, disposição em ordem dos elementos essenciais que compõem um corpo (concreto ou abstrato)
- Estrutura
- Modelos de desenvolvimento e resolução
- Objetivo:
  - Escrita
  -Entendimento
  - Validação
  -Manutenção

**3. Técnica Modular**

- Partes independentes controlada por um conjunto de regras

# 3.Fundamentos de algoritmos

## 3.1 Tipologia e variáveis

A informações passadas ao computador são de dois tipos:

- Dados: manipulação; tratados e processados.
- Instruções: executam ações que processam os dados; operações.

**Dados:**

Numéricos

- Inteiros - 0,1,5,50,800.../-58, -50, -49, 32
- Reais - 5.95, 9.54, -8.8,

Caractere

- Tudo aquilo que não representamos como números (mas números também são)
- "KU*&NH53g'

Lógico - Booleano

- Verdadeiro - 1
- Falso - 0

**Variável**
Um tipo de estrutura mutável, inconstante. Pode assumir qualquer um dos valores de um determinado conjunto de valores.

Boas práticas

- Atribuição de uma ou mais caracteres - nome que dê sentido à variável
- Primeira letra - não número
- Sem espaços em branco
- Vetado - utilização de palavras reservadas
- Caracteres e números

Papéis

- Ação -  modifica o estado do algoritmo/programa
- Controle - vigiada, usada para controlar uma estrutura

**Constante**
Algumas variáveis são definidas como/por constantes, como por exemplo estruturas que tem seu valor fixo. Ex.: pi = 3,14; phi = 1,618

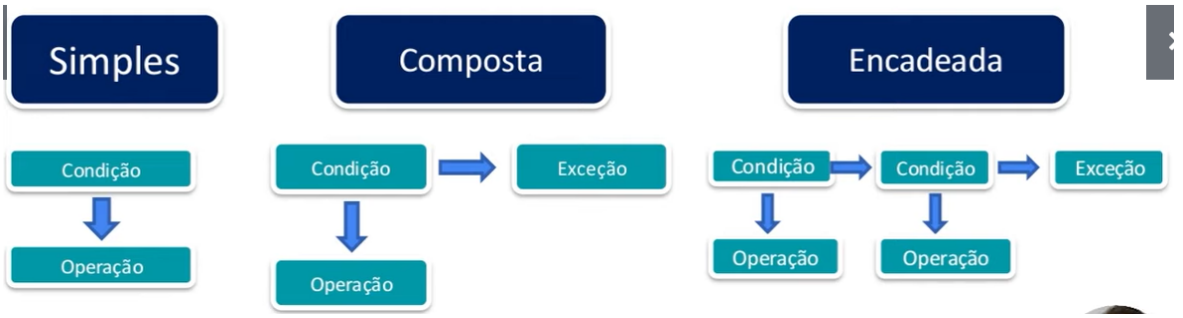
## 3.2 Instruções primitivas

- Calculos matemáticos

| Operador | Operação | Tipo | Prioridade Matemática | Tipo de Retorno de Resultado |
|---|---|---|---|---|
| + | Manutenção de sinal | Unário | 1 | Positivo |
| - | Inversão de sinal | Unário | 1 | Negativo |
| ↑ | Exponenciação | Binário | 2 | Inteiro ou real |
| / | Divisão | Binário | 3 | Real |
| div | Divisão | Binário | 4 | Inteiro |
| * | Multiplicação | Binário | 3 | Inteiro ou real |
| + | Adição | Binário | 4 | Inteiro ou real |
| - | Subtração | Binário | 4 | Inteiro ou real |

## 3.3 Estruturas condicionais

Estado de uma pessoa ou coisa; expressa uma suposição. Há um condição que se satisfeita, executa uma determinada instrução.




**Operadores relacionais**

| Símbolo | Significado |
|---|---|
| = | Igual a |
| <> | Diferente de |
| > | Maior que |
| < | Menor que |
| >= | Maior ou igual a |
| <= | Menor ou igual |

**Condicional Simples**

Se (<condição>) então

<instruções para condição verdadeira>

fim_se

**Condicional Composta**

Se (<condição>) então
    <instruções para condição verdadeira>
Senão
    <instruções para condição falsa>
fim_se

**Condicional encadeado**
Se(<condição 1>) então
    <instruções para condição verdadeira>
Senão
    Se(<condição2>) então
        <instruções para condição2 verdadeira e condição1 falsa>
        Senão
        <instruções para condição 1 e 2 falsas>
fim_se

**Operadores lógicos**

- Verificação de V ou F
- Substituição - encadeamento de condições

**And**
Condição verdadeira - todas devem ser satisfeitas
**Or**
Apenas uma das condições deve ser verdadeira
**Not**
Inversão do resultado lógico

## 3.4 - Estruturas de repetição

Laço, controle de fluxo, loop, malhas de repetição

Condição de parada

- Número de repetições pré-fixada
- Condição a ser satisfeita

Vantagens

- Redução de linhas
- Compreensão facilitada
- Redução de erros

### 3.5 - Vetores e Matrizes

- Vetor - "variável dimensionada com tamanho pré-fixado", unidimensional
- Matriz - "tabela organizada em linhas e colunas no formato m(número de linhas horizontal) x n(número de colunas, vertical)".
  - Coleção de variáveis
  - Contíguas em memória
  - Índices

### 3.6 - Funções

Similar ao conceito de função matemática; vai implicar que um objeto A esteja concectada á um conjuno B.
"Funções ou sub-rotinas são blocos de instruções que realizam tarefas específicas."
Trechos de códigos que são chamados em um programa para realizar tarefas, identificados por nomes e parâmetros.

- Modularização do programa
- Código mais claro e conciso
- Reutilização de instruções

### 3.7 - Instruções de entrada e saída

- Entrada: inserção e recebimento de dados do mundo real por meio de ação de alguma interface, seja teclado, mouse, arquivos, entre outros.
- Saída: impressão dos dados do mundo abstrato, digital po meio de ação de alguma interface.

Os formatos pordem variar desde simples arquivos binários até complexas querys de banco de dados.

## 4. Linguagens de Programação

### 4.1 - Introdução

Método padronizado composto por um conjunto de regras sintáticas e semânticas de implementação de um código fonte.

**Código fonte**

- Traduzido: linguagem de alto nível (programa fonte) > compilador (executa análise do programa) > linguagem de máquina (programa objeto)
  (1) Geração do programa objeto
  (2) Execução do programa objeto
  execução mais rápida e programas menores
- Interpretado: Programa fonte executado diretamente
  maior flexibilidade

**Características de um programa**

- Legibilidade
  - Facil de ler
  - Compreensão
  - Ortogonalidade - coerência nas instruções
  - Definição adequada das estruturas
- Redigibilidade
  - Facilidade de escrita de código
  - Simplicidade de escrita
  - Suporte à abstração
  - Reuso do código
  - Expressividade
- Confiabilidade
  - Faz o que foi programado para fazer
  - verificação de tipos
  - uso de ponteiros
  - compatibilidade entre compiladores
- Custo
  - Análise de impacto
  - Treinamento
  - Compilação
  - Execução
  - Infra-estrutura

- Outras
  - Atualizações
  - Uso para IA
  - Disponibilidade de ferramentas
  - Comunidade ativa
  - Adoção pelo mercado

**Análises de Código**

- Análise léxica: particionar > classificar > eliminar
- Análise sintática: forma; componentes que interligam os constituintes da sentença, atribuindo-lhe uma estrutura; corretudo do programa
- Semântica: significado; relação entre significantes, como palavras, frases, sinais e símbolos.

**Paradigmas de Programação**

- **"**Forma de resolução de problemas com diretrizes e limitações específicas de cada paradigma utilizando linguagem de programação".

*Estruturado (Linguagem em C)*

- Sequência
- Decisão (teste lógico)
- Iteração (funções, laços, condições)
- Utilizado em problemas simples e diretos, aprender programação.

*Orientação à Objeto*

- Baseado na utilização de objetos e suas interações
- Análogo ao mundo real
- Objeto: é descrito por características específicas, comportamentos e estado. "O que tenho, o que sou capaz de fazer, como faço."