



Dio - Bootcamp

▼ Introdução ao Desenvolvimento Moderno de Software

[Desenvolvimento Moderno de Software.pdf](#)

Mas o que são os frameworks?

Framework é, de forma básica, um facilitador.

Ele traz diversas soluções já pré-definidas, que descomplicam o trabalho dos profissionais no desenvolvimento de aplicativos e outros projetos digitais.

Afinal, a atuação de um programador pode ter muito de criatividade, mas também traz aspectos mecânicos, de repetição de tarefas, que seriam maçantes sem a possibilidade de automatização.

Exemplos: Angular, Laravel e Vue



Nuvem Pública

Os sistemas em nuvem são sistemas de armazenamento de dados disponibilizados via internet, em vez de servidores físicos tradicionais. Hoje, muitas organizações estão migrando o armazenamento de dados de servidores físicos para sistemas baseados em nuvem.



▼ Introdução à Programação e Pensamento Computacional

<https://drive.google.com/file/d/1vemC6G790JNte1882V53DAKEawJzVIWL/view?pli=1>

▼ Pensamento Computacional

ChatGPT:

O pensamento computacional é uma abordagem para resolução de problemas que envolve habilidades e conceitos fundamentais utilizados pelos cientistas da computação. Ele envolve a aplicação de estratégias e técnicas de resolução de problemas que são comumente usadas no campo da computação. Aqui estão os princípios-chave do pensamento computacional:

1. Decomposição: Consiste em dividir um problema complexo em partes menores e mais gerenciáveis. A decomposição permite que você aborde cada parte separadamente, facilitando a compreensão e a resolução do problema como um todo.
2. Reconhecimento de padrões: Envolve a identificação de padrões e regularidades em dados, informações ou problemas. Ao reconhecer padrões, você pode aproveitar soluções existentes e aplicá-las a problemas semelhantes.
3. Abstração: Trata-se de simplificar um problema, concentrando-se apenas nos aspectos mais relevantes e ignorando detalhes desnecessários. A abstração permite uma compreensão mais clara e uma solução mais eficiente.
4. Algoritmos: São sequências de instruções claras e passos definidos para resolver um problema. O pensamento computacional envolve a capacidade de projetar e implementar algoritmos eficientes para resolver problemas específicos.
5. Pensamento algorítmico: É a capacidade de formular problemas de maneira que possam ser resolvidos por algoritmos. Envolve a capacidade de identificar a sequência correta de etapas e tomar decisões lógicas para resolver um problema.

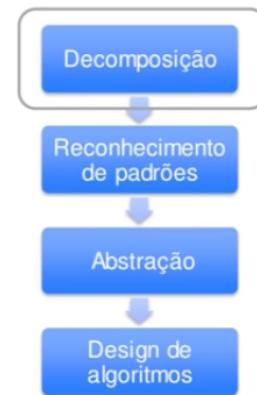
6. Automação: Consiste em usar a tecnologia para automatizar tarefas repetitivas ou demoradas. O pensamento computacional incentiva a automação para aumentar a eficiência e reduzir erros.
7. Generalização: Trata-se de aplicar soluções e conceitos aprendidos em um contexto específico a problemas diferentes, mesmo em áreas não relacionadas. A generalização permite que você adapte soluções existentes para resolver novos problemas.
8. Experimentação: Envolve a tentativa e erro, a exploração de diferentes abordagens e a capacidade de ajustar soluções com base nos resultados. A experimentação é uma parte essencial do pensamento computacional para iterar e melhorar continuamente as soluções.

Esses princípios do pensamento computacional podem ser aplicados não apenas no campo da ciência da computação, mas também em diversas áreas, como matemática, ciências, engenharia e até mesmo no cotidiano, como uma abordagem sistemática para resolver problemas complexos de forma eficiente.

Overview

Baseado em 4 pilares

Dividir um problema complexo em subproblemas



[21]

Pilares: Decomposição



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

Dado um problema complexo, devemos quebra-lo em problemas menores. Portanto, problemas mais fáceis e gerenciáveis.

Pilares: Padrões



Reconhecimento de padrões

- Estrutura base
- Estrutura invariante

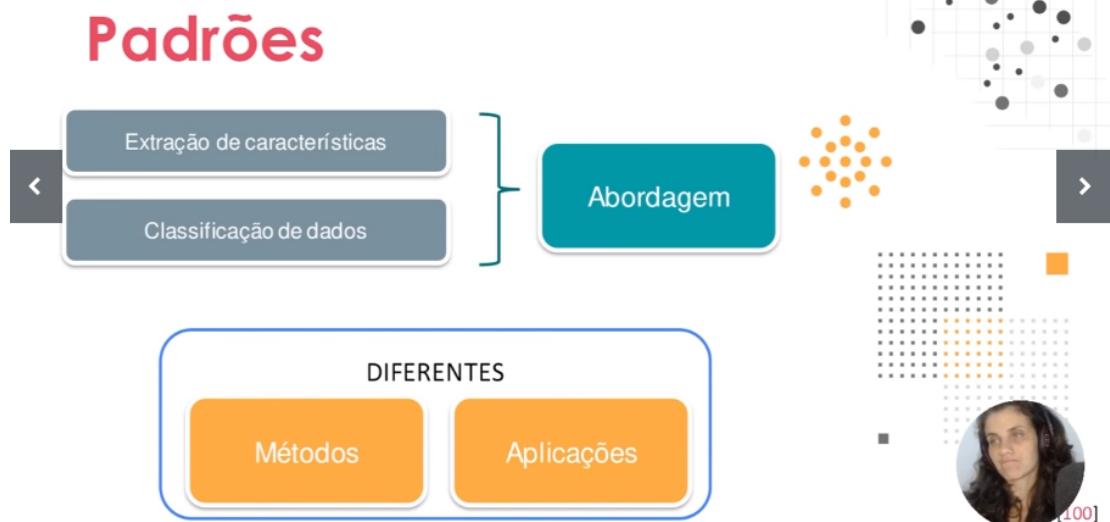
- Repetições

Similaridades e diferenças

Por que determinar padrões?

Generalizar, com o objetivo de obter resoluções para problemas diferentes

Um computador reconhecer padrões através de comparação, o ser humano é subjetivo, mas o computador ele é objetivo e direto, é preciso passar dados e informações para ele, para que ele seja capaz de fazer uma comparação.



Pilares: Abstração

Generalização / Abstração

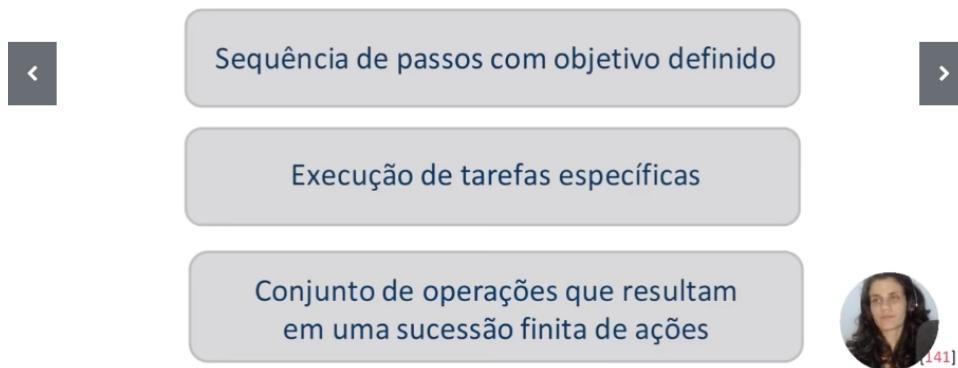
- **Abstrair**
 - Observar, um ou mais elementos avaliando características e propriedades em separado.
- **Abstração**
 - Processo intelectual de isolamento de um objeto da realidade
- **Generalizar**
 - Tornar-se geral, mais amplo, extensão

Pilares: Algoritmos

Um programa é constituído por uma série de instruções, um algoritmo são instruções passo a passo que diz para o computador o que ele tem que fazer e qual é a ordem desse processo, são essas instruções, algoritmos que são utilizadas para resolver um problema.



Algoritmos

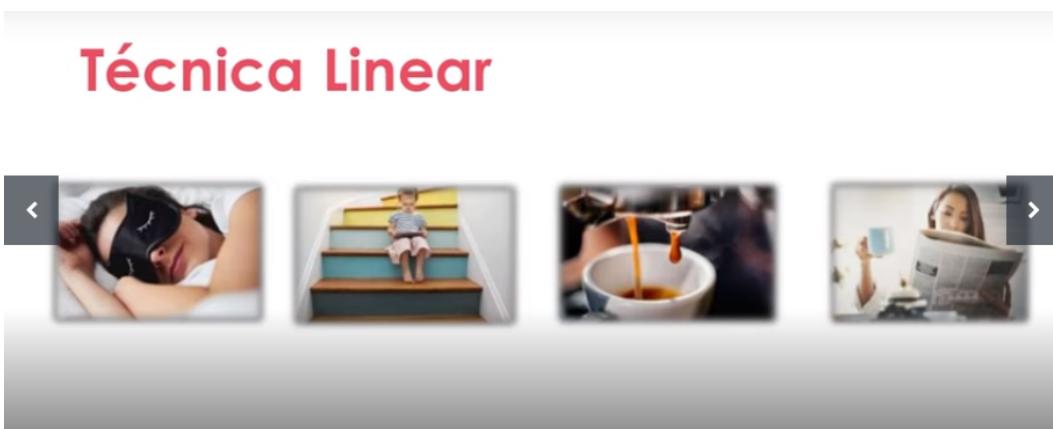


▼ Introdução a lógica de programação

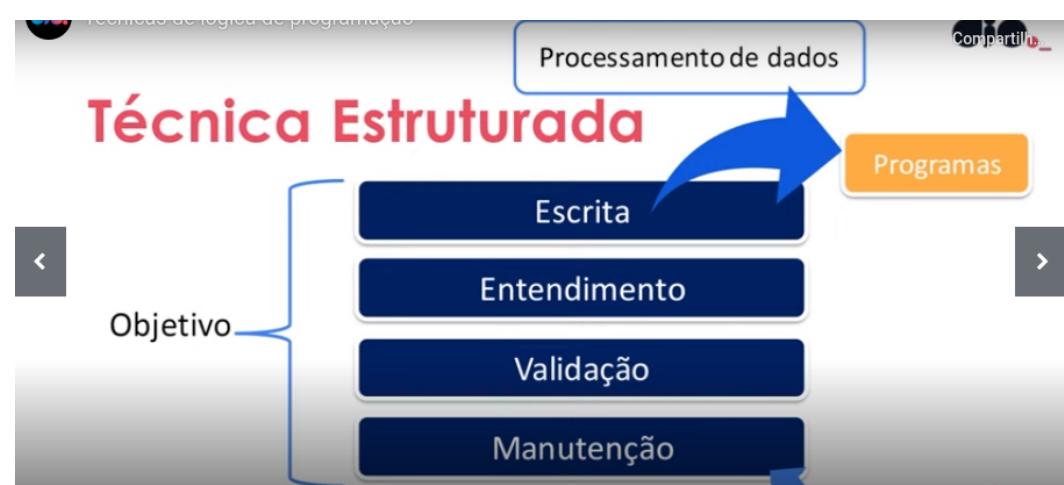
Organização e planejamento das instruções, assertivas em algoritmo, a fim de viabilizar a implementação de um programa.

Técnicas de lógica de programação

Técnica linear



Técnica estruturada



Técnica Estruturada

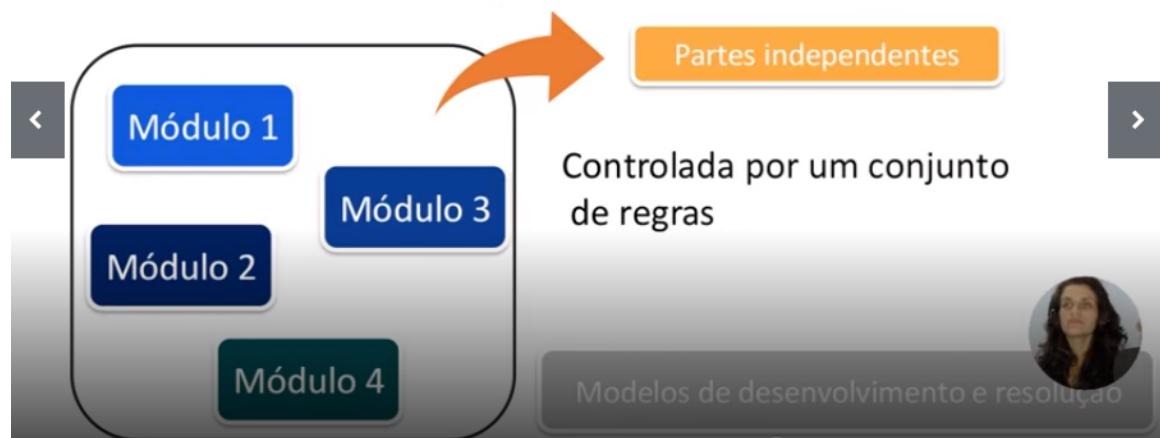


[224]

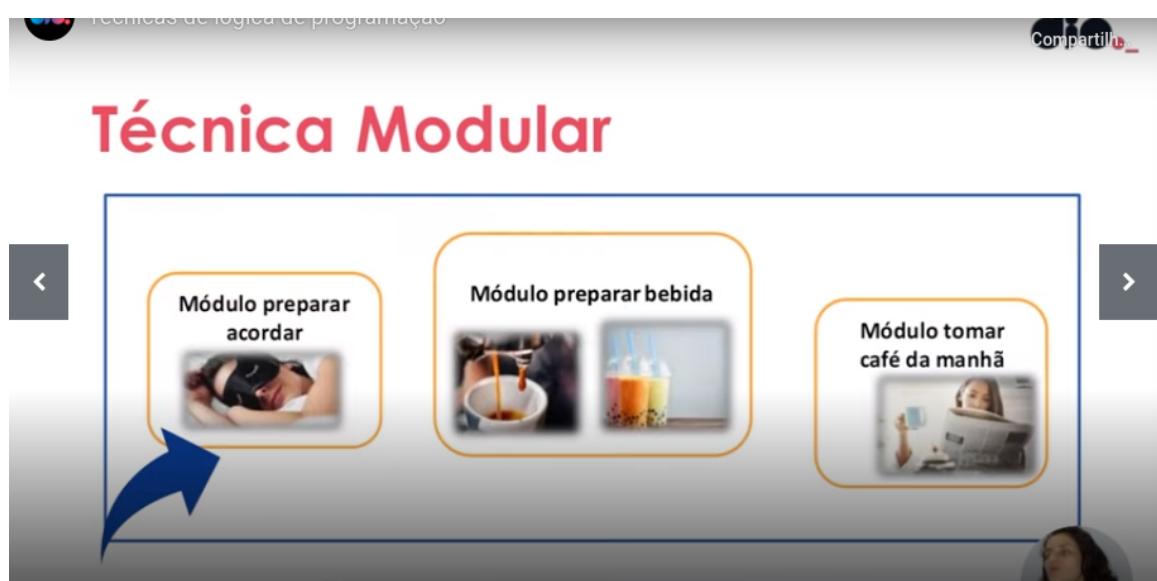
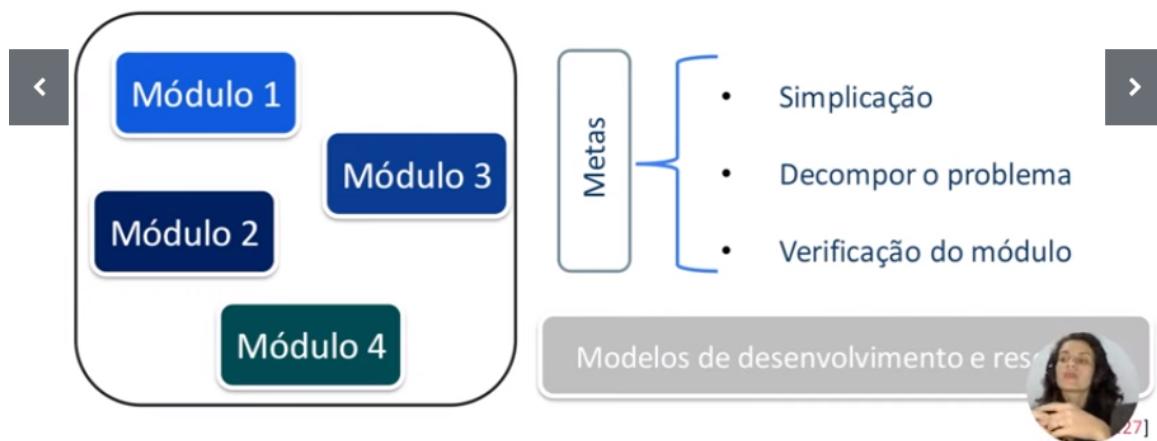
Técnica Modular

Parte independentes controladas por um conjunto de regras.

Técnica Modular



Técnica Modular

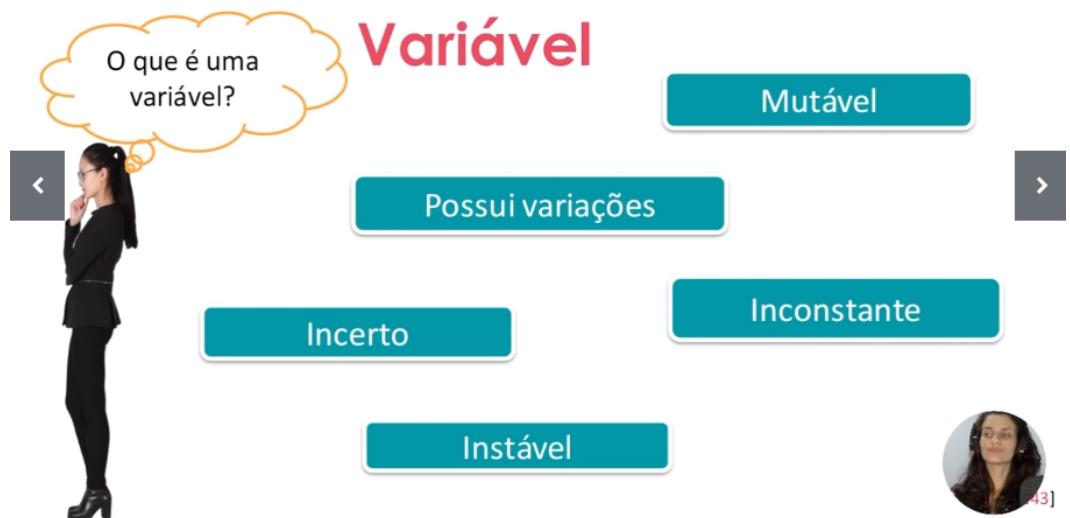


▼ Fundamentos de algoritmos

Tipologia e variáveis

A função do computador é processar as informações passada para ele, essas informações são Dados como: Number, String, Booleano e as instruções são os processo de como esses dados deverão ser processados.

Variáveis



Variável

Identificar

Variável



Variável

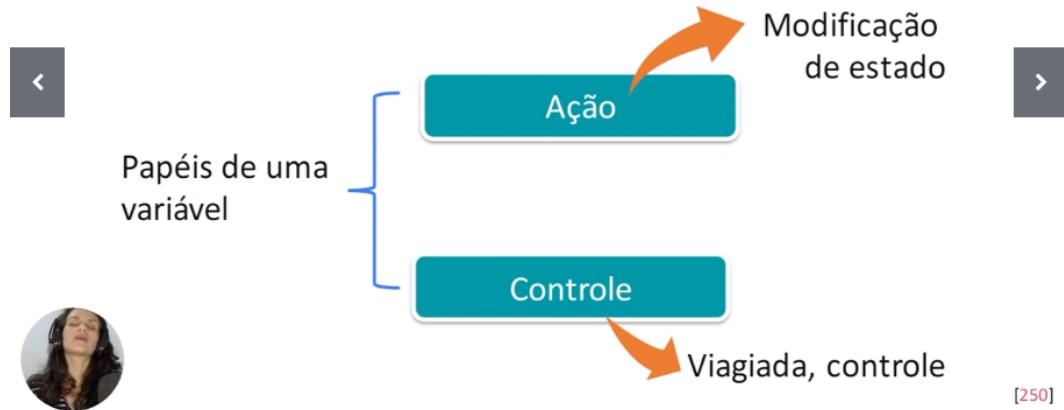
Nome da variável

Regras

- Atribuição de um ou mais caracteres
- Primeira letra - não número
- Sem espaços em branco
- Vedado
 - Utilização de palavras reservadas
- Caracteres e números



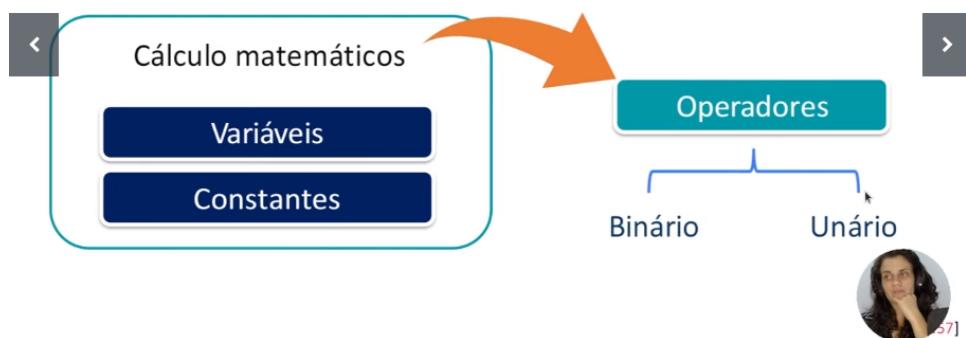
Variável



Instruções primitivas

As instruções iram determinar as ações que iremos executar em cima dos nossos dados, essas ações geralmente são cálculos matemáticos e para executar esses cálculos utilizamos operadores.

Instruções Primitivas



Instruções Primitivas

Operadores

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

Instruções Primitivas

Definição formal

Instruções são linguagem de **palavras-chave** (vocabulário) de uma determinada de programação que tem por finalidade comandar um computador que irá **tratar os dados**

O que são instruções?

Exemplo

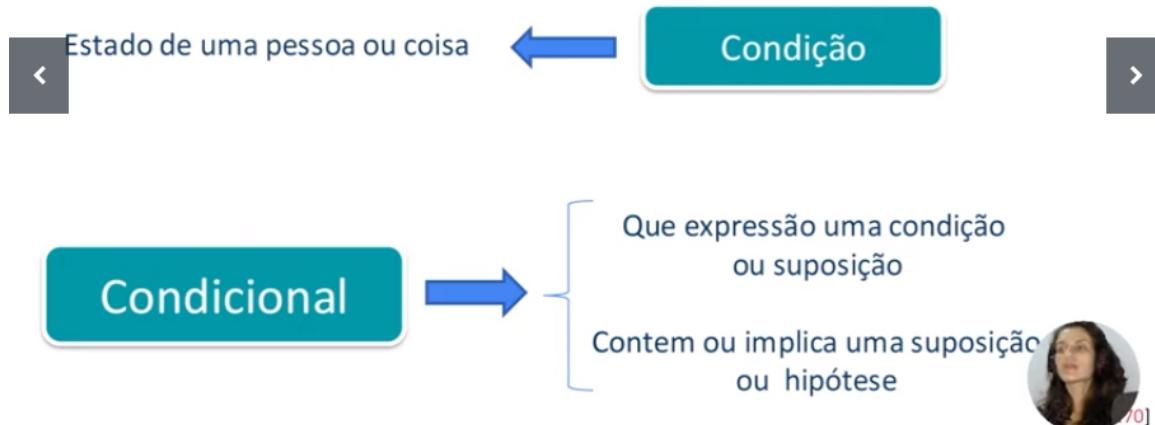
Média escolar

Estrutura sequencial

```
Início program
Variável
Nota1 = 7
Nota2 = 8
Resultado = 0
Resultado = (Nota1 + Nota2)/2
Constante
Escreva resultado
Fim programa
Saída:
6.5
```

Estruturas condicionais e operadores

Estrutura Condicional



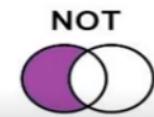
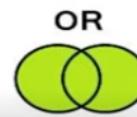
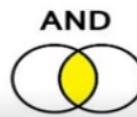
Estrutura Condicional

Operadores Relacionais

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

Estrutura Condicional

Operadores Lógicos



Fonte: biblio.info

Estruturas de repetição

Estruturas de Repetição

Trecho de um programa

Laços

Controle de fluxo

Malhas de repetição

Repetição

Loop

Estruturas de Re

Trecho de um programa

Redução de linhas

Compreensão facilitada

Redução de erro

Vetores e matrizes

Vetores e Matrizes

Container

Definição formal

"Um vetor é caracterizado por uma variável dimensionada com tamanho pré-fixado."

Matriz unidimensional

O que são funções?

Funções

Subprograma

Subalgoritmo

Função

Bloco

Método

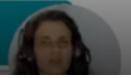
Sub-rotina

Funções

Definição formal

As funções, ou sub-rotinas são blocos de instruções que realizam tarefas específicas

Decomposição do algoritmo



A assinatura da função é determinada pelo nome da função e o parâmetro que ela possuí

Funções

Função? Não é mais fácil fazer tudo junto?

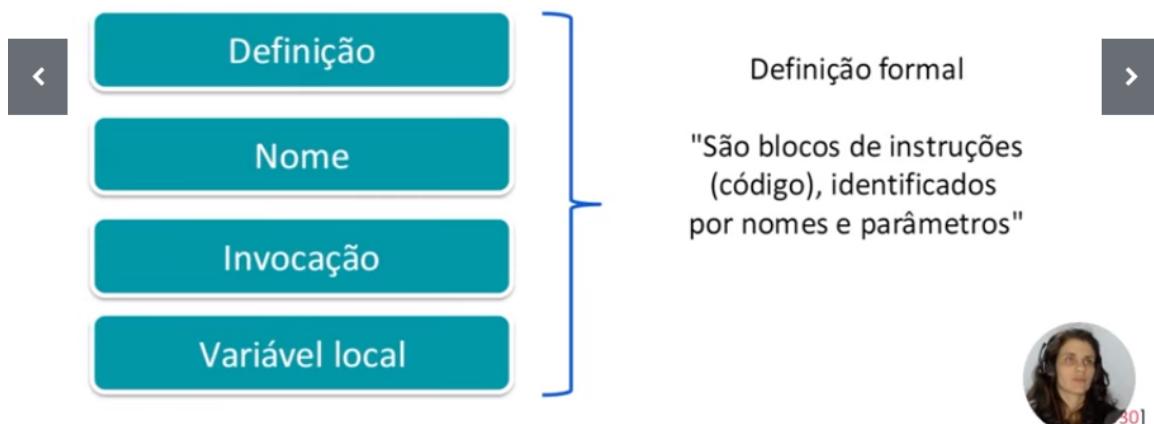
Definição formal

"São blocos de instruções (código), identificados por **nomes e parâmetros**"

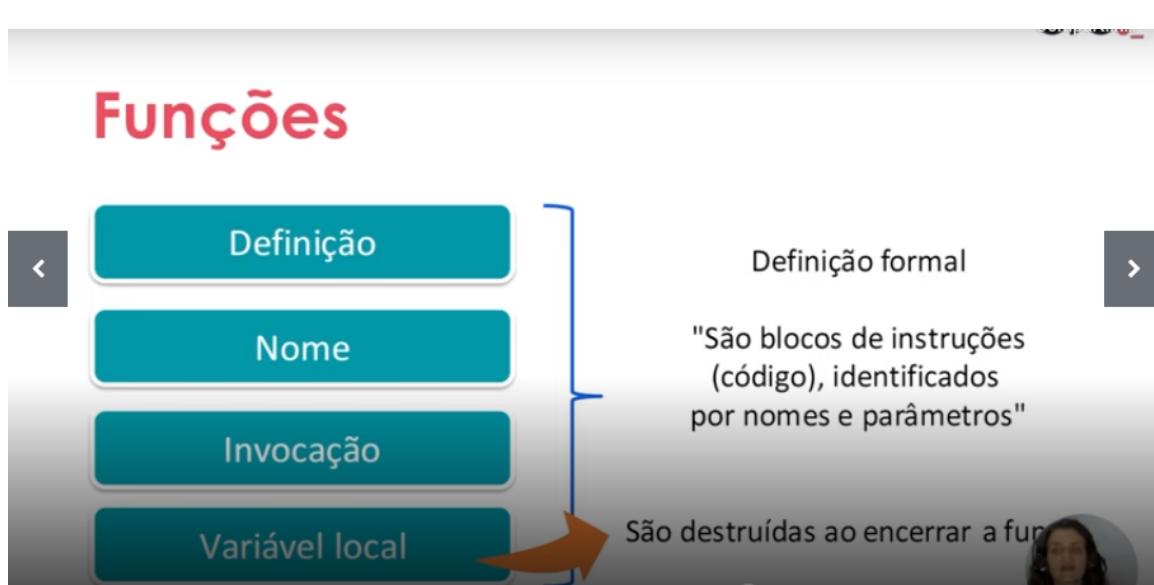
Assinatura



Funções



ATENÇÃO: As variáveis dentro da função só existem enquanto ela estiver em execução, no momento que todas as instruções dentro da função for executada e ela retornar um valor ou não, as variáveis que estiverem dentro da função serão destruídas, ou seja, serão desalocadas da memoria, não existirão mais, serão apagadas. Mas esse comportamento irá depender de cada linguagem já que cada uma tem um comportamento diferente.



Instruções de entrada/saída

Instruções de entrada/saída

Definição formal



Consiste na inserção e recebimento de dados do mundo real por meio de ação de alguma interface, seja teclado, mouse, arquivos, entre outros.



Instruções de entrada



Instruções de entrada/saída

Definição formal



Consiste na impressão dos dados do mundo abstrato, digital por meio de ação de alguma interface.



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

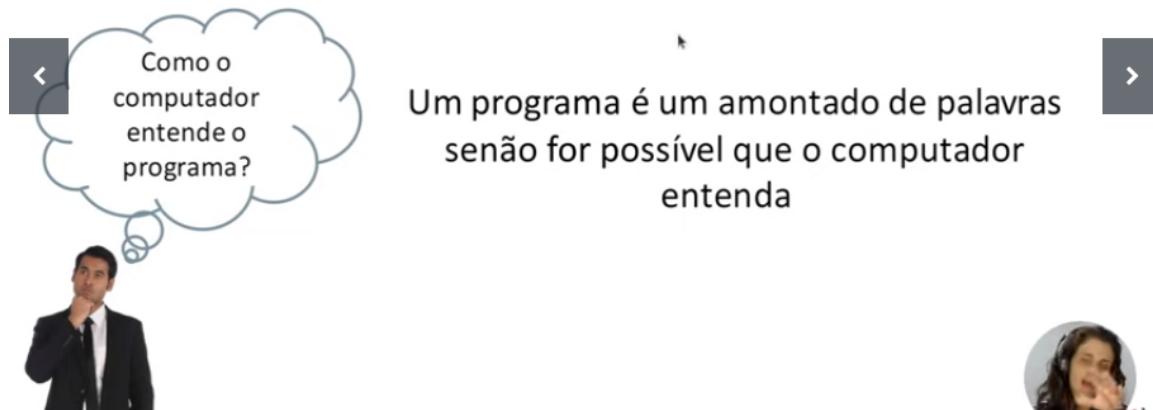
Instruções de saída



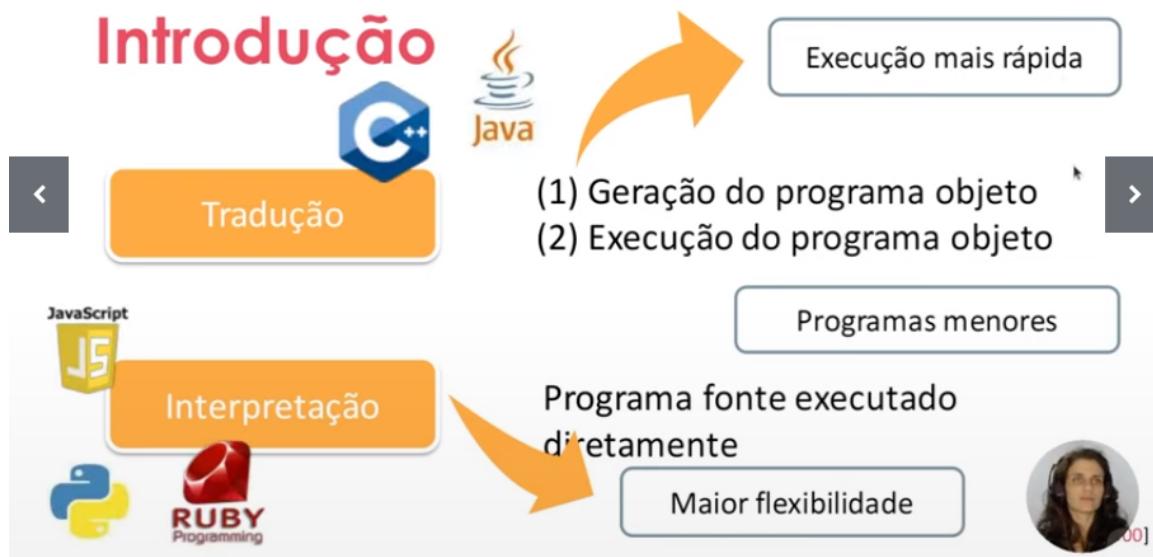
▼ Linguagens de programação

Como um computador entende o programa?

Introdução



dio





Se uma computador não entende as instruções de um programa, então são apenas um amontoado de palavras, e como o computador faz para entender o que programamos?

Existem dois tipos de linguagem a de baixo nível e a de alto nível, as linguagens de programação de alto nível são aquelas linguagem que estão mais próximas da nossa linguagem, com uma sintaxe parecida, por isso, são mais fáceis de entendermos, agora as linguagens de baixo nível são linguagens de máquina, ela tem esse nome porque a sua sintaxe está mais próxima da linguagem da máquina, o ponto negativo é que por causa disso é mais difícil para entendermos.

Por causa disso, tem linguagem que são traduzidas ou interpretadas:

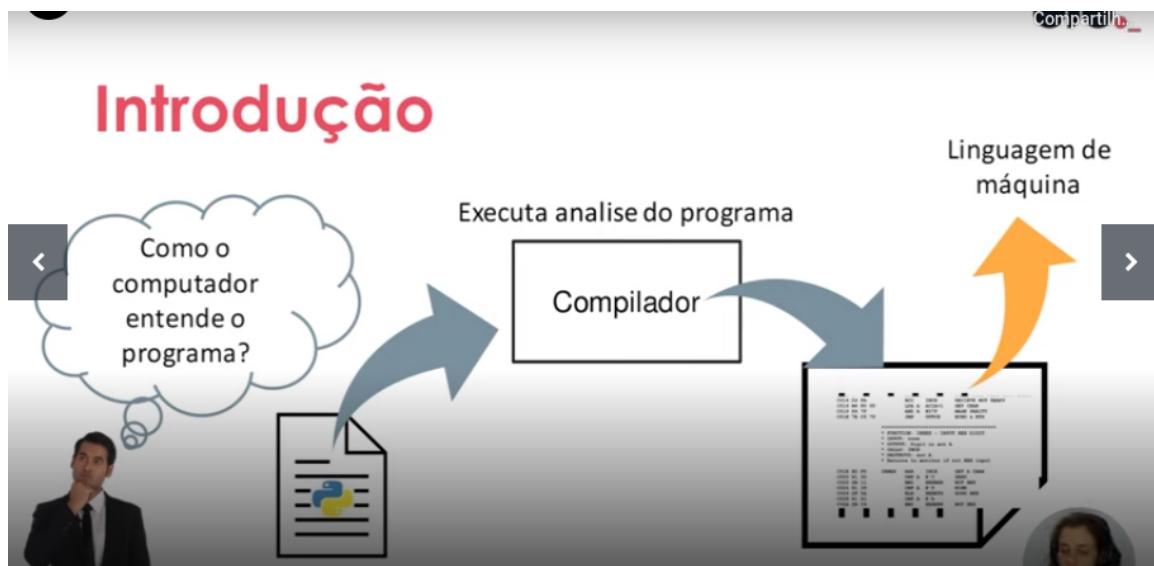


Traduzida:

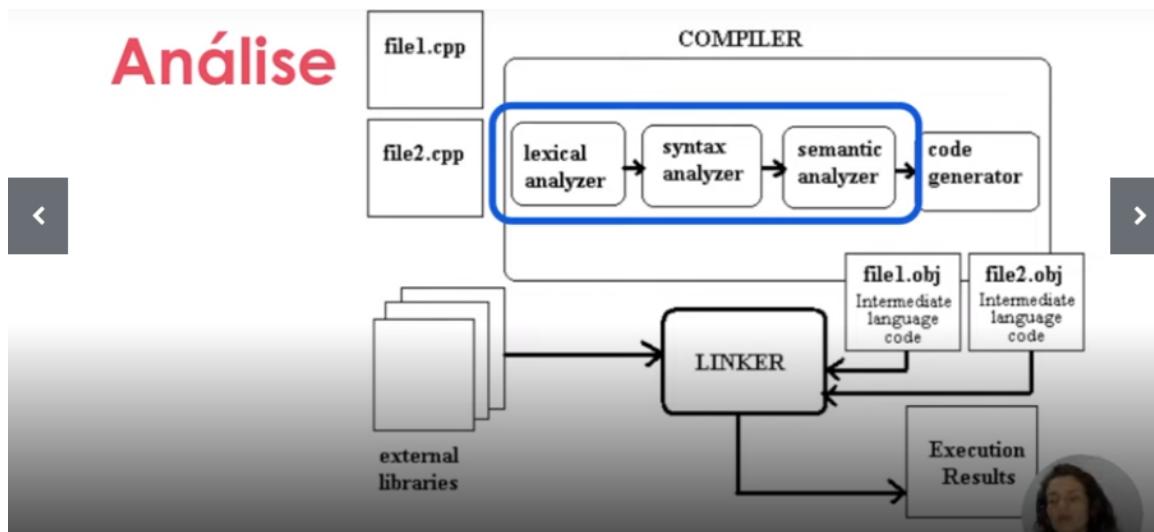
- Uma linguagem que é traduzida tem o seu programa fonte que é a linguagem na qual você programou enviada para um compilador, esse compilador irá traduzir essa linguagem de alto nível para uma linguagem de baixo nível, gerando no final um programa objeto, esse programa objeto contém as instruções em linguagem de máquina que será executada pelo computador.
- Ponto positivo: por traduzir a linguagem de alto nível para uma de baixo nível, a execução é mais rápida, já que o computador está executando o programa objeto que contém as instruções em linguagem de máquina, e por essa linguagem estar mais próxima da linguagem da máquina a sua leitura é mais rápida. **Exemplo a linguagem Java que é uma linguagem compilada.**

Interpretada:

- Uma linguagem interpretada é uma linguagem de alto nível, e invés de ser compilada ela é interpretada, então não será gerado um programa objeto, e sim o computador irá entender essa instruções de alto nível através de um interpretador, por causa disso, a leitura é mais devagar, ou seja, uma linguagem que é compilada é mais rápida que uma linguagem interpretada como por exemplo a linguagem Javascript.



Análises de código



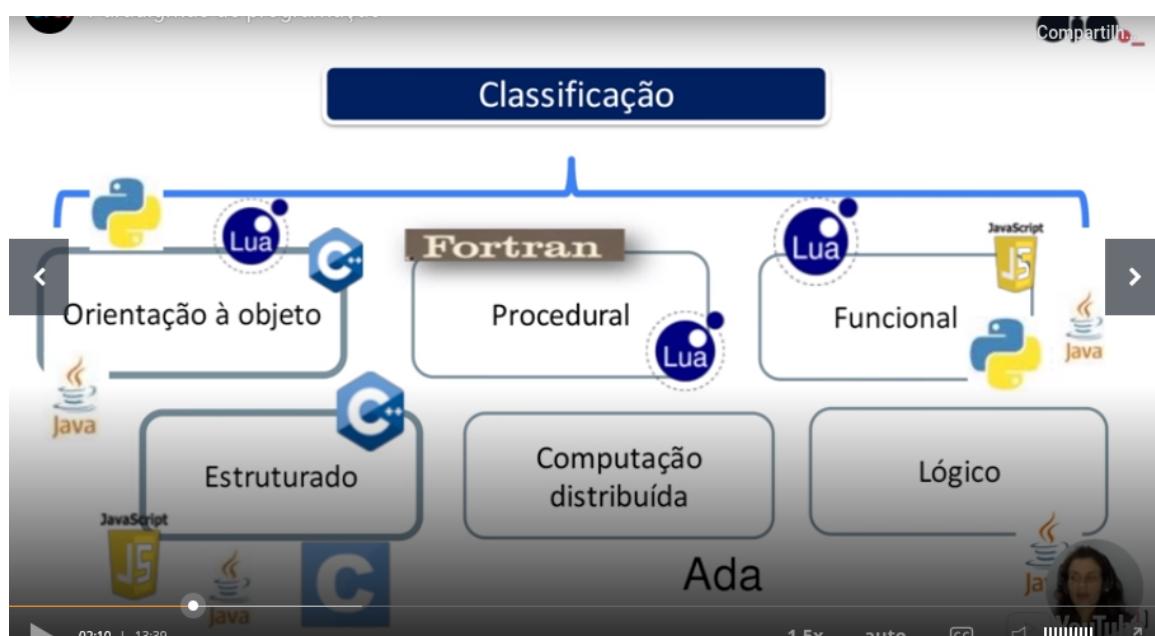
Paradigmas de programação

Paradigmas

O que é um paradigma?

Definição formal

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



Paradigma Estruturado

Conceitos:

- Sequência
- Decisão
- Iteração



Ênfase em sequência

Orientação à Objeto

Paradigma de programação
baseado na utilização de objetos
e suas interações

Análogo ao mundo real



Orientação à Objeto

O que é um objeto?

Ponto de vista da programação

The diagram illustrates the comparison between Object-Oriented Programming (POO) and Structured Programming (Estruturada). It features two main boxes: 'POO' on the left and 'Estruturada' on the right. The 'POO' box contains a rounded rectangle labeled 'Classe' with two sub-boxes: 'Alocação em memória' and 'Operações associadas'. The 'Estruturada' box contains a rounded rectangle labeled 'Variável' with two sub-boxes: 'Alocação em memória' and 'Operações associadas'. Arrows point from the sub-boxes in the 'Classe' box to their corresponding boxes in the 'Variável' box, indicating a mapping or relationship. A bracket groups the 'Alocação em memória' and 'Operações associadas' boxes under both 'POO' and 'Estruturada'.

09:56 | 13:39

1.5x auto CC

▼ Git e GitHub

O Git é um sistema de versionamento de código distribuído, ele é uma ferramenta de versionamento de código porque permitir rastrear as alterações e mudanças feita no código ao longo do tempo criando no final um histórico de mudanças, uma linha do tempo e o Git é um sistema de código distribuído porque permiti que cada desenvolvedor tenha uma cópia do repositório original no seu repositório local.

Ciclo de vida dos arquivos Git



O repositório local só vai ser composto por **commits**, tudo que estiver no seu repositório local tem que estar commitado, se não, não terá como você fazer um **push** para o seu repositório remoto.