

# Primeiros passos para começar a programar

*Curso base para iniciante*

**Juliana Mascarenhas**

Tech Education Specialist / Sócia (Content Creator) @SimplificandoRedes

Me Modelagem Computacional / Cientista de dados

**@in/juliana-mascarenhas-ds/**



<https://github.com/julianazanelatto>

# Juliana Mascarenhas

Tech Education Specialist

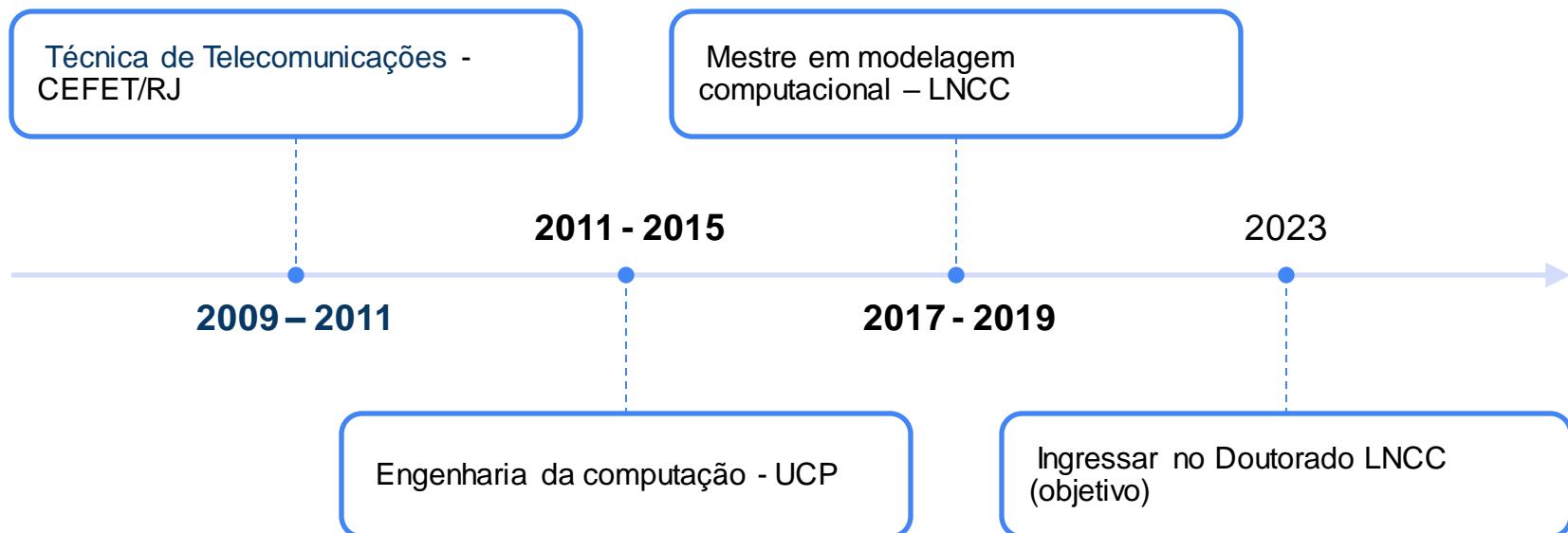
Sócia no @SimplificandoRedes

Cientista de dados

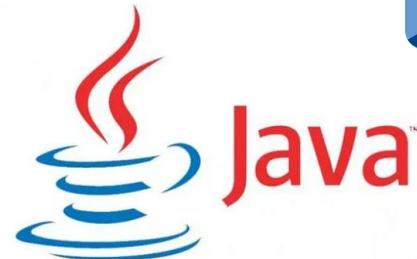
Desenvolvedora Java/Python

Me Modelagem Computacional - LNCC

# Sobre Mim



# Sobre Mim



PostgreSQL



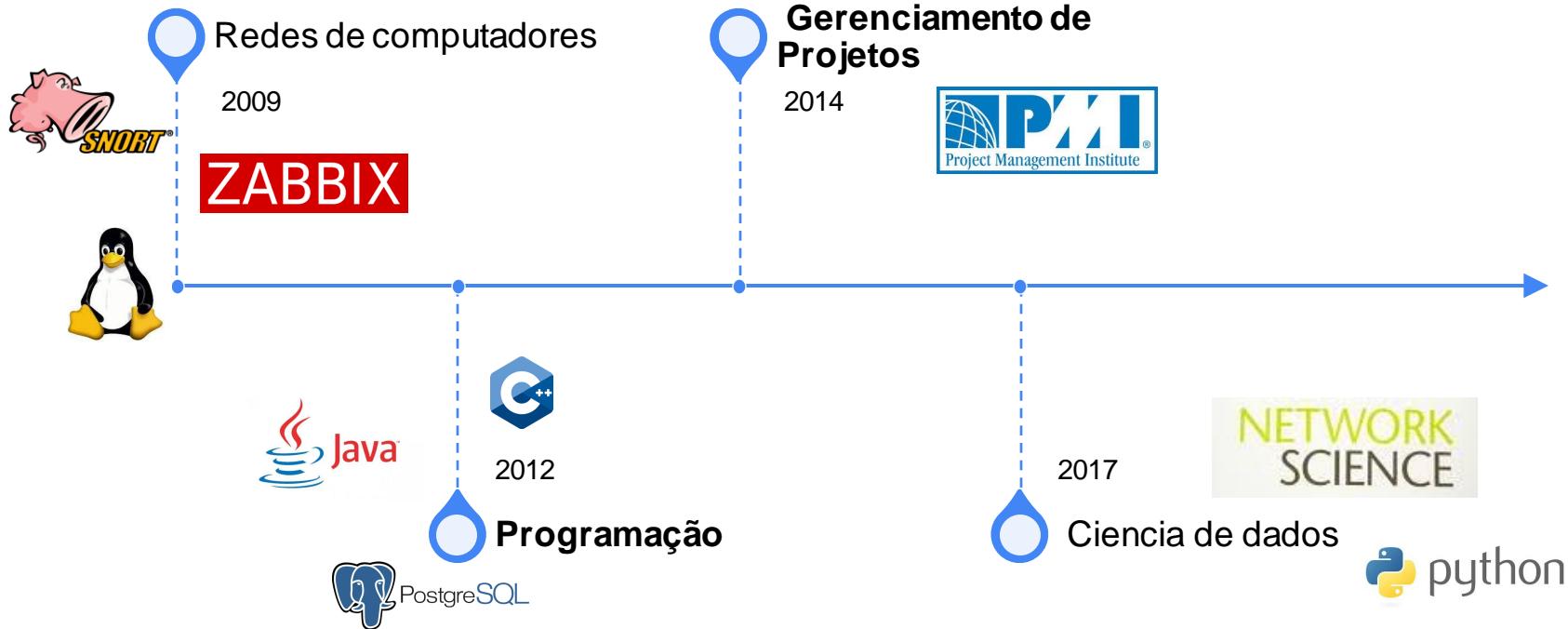
APACHE



ZABBIX



# Sobre Mim



# Cursos



## Criando uma API REST conectada a Amazon RDS com Java

Juliana Mascarenhas  
Mestre em Modelagem Computacional

Modelagem Computacional para Desenvolvedores



## Java Spring com RabbitMQ

Juliana Mascarenhas  
Mestre em modelagem computacional

Modelagem Computacional para Desenvolvedores

## Criando um Microserviço de Upload de Imagens com o Amazon S3

Juliana Mascarenhas  
Mestre em Modelagem Computacional

Modelagem Computacional para Desenvolvedores

# Objetivo Geral

Você será capaz de entender o que significa pensar computacionalmente.

Pensamento aplicável à qualquer área do conhecimento.

Dessa forma, será capaz de resolver qualquer problema de uma maneira mais objetiva e eficiente.

# Percorso

Aula 1

Pensamento Computacional

Aula 2

Introdução à lógica da programação

Aula 3

Fundamentos de Algoritmos

# Percorso

Aula 4

Linguagens de programação

Aula 5

Primeiro contato com a programação

# Aula 1: Pensamento Computacional

// Primeiros passos para começar a programar

# Objetivo Geral

Apresentar os conceitos que caracterizam o pensamento computacional, permitindo que o Dev entenda o que significa pensar computacionalmente.

# Percurso

**Etapa 1**

**Introdução ao pensamento computacional**

**Etapa 2**

Habilidade complementares

**Etapa 3**

Pilares: Decomposição

# Percurso

Etapa 4

Pilares: Padrões

Etapa 5

Pilares: Abstração

Etapa 6

Pilares: Algoritmos

# Percurso

Etapa 7

Estudo de caso conceitual: perdido

Etapa 8

Estudo de caso aplicado: Soma de um intervalo

Etapa 9

Estudo de caso aplicado: Adivinhe um número

## Etapa 1

# Introdução ao pensamento computacional

// Primeiros passos para programar/Pensamento  
Computacional

# Overview

Pensamento computacional?

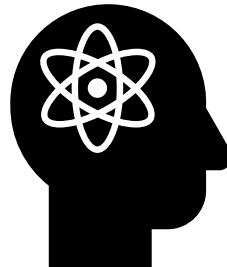


Processo de pensamento envolvido na expressão de soluções em passos computacionais ou algoritmos que podem ser implementados no computador.

(Aho, 2011; Lee, 2016)

# Overview

Pensamento computacional?

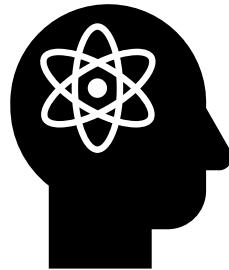


Sistemático e eficiente

Formulação e resolução de problemas

# Overview

Pensamento computacional?



Sistemático e eficiente

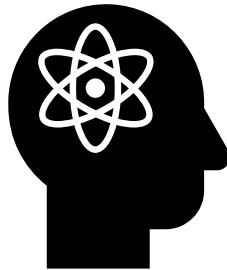
Formulação e resolução de problemas

Sejam capazes de resolver

Humanos & máquinas

# Overview

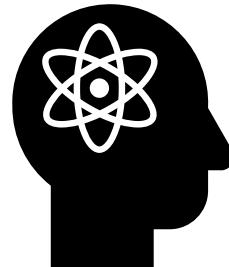
Pensamento computacional?



Habilidade  
generalista

# Overview

Pensamento computacional?



Matemática

Leitura

Escrita

Disciplina  
acadêmica

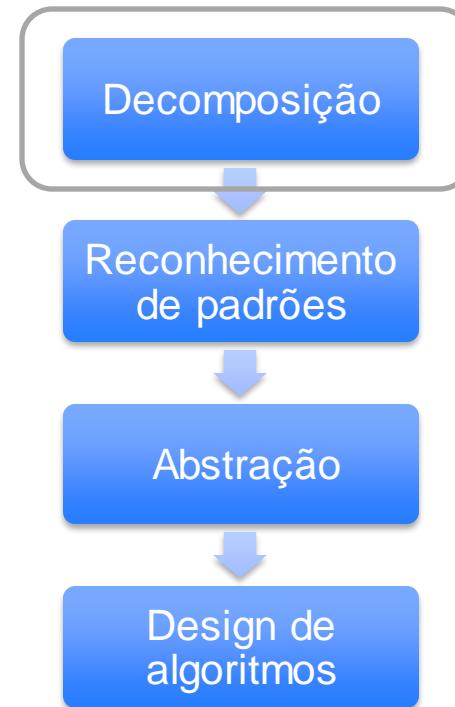
Habilidade  
generalista



# Overview

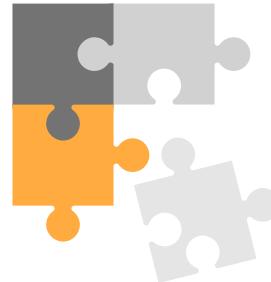
Baseado em 4 pilares

Dividir um problema complexo em subproblemas



# Overview

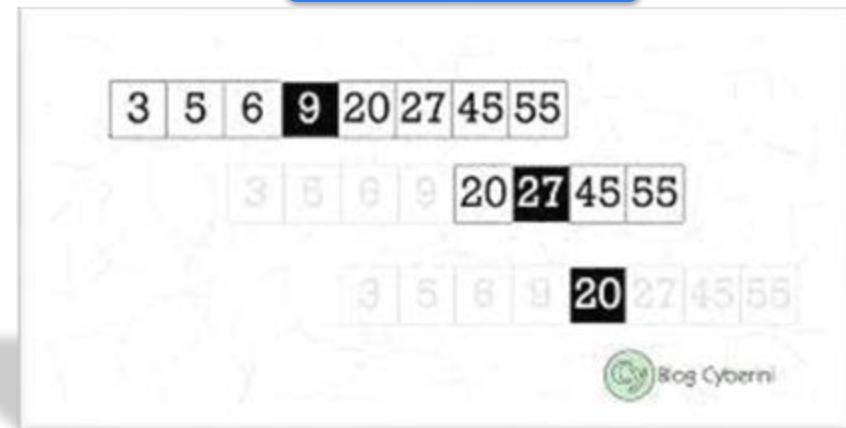
Baseado em 4 pilares



Decomposição

Reconhecimento  
de padrões

Dividir um problema  
complexo em subproblemas



# Overview

Baseado em 4 pilares

Identificar padrões ou tendências

Decomposição

Reconhecimento de padrões

Abstração

Design de algoritmos

# Overview

Baseado em 4 pilares

Identificar padrões ou tendências

Similaridades e diferenças entre os problemas



Decomposição

Reconhecimento de padrões

Abstração

Design de algoritmos



# Overview

Baseado em 4 pilares

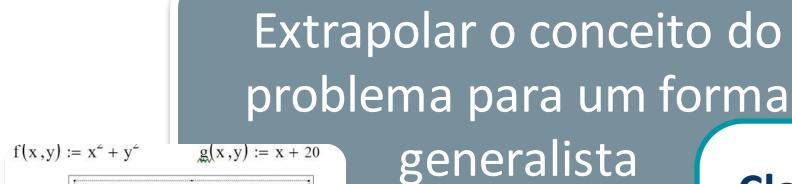
Extrapolar o conceito do problema para um forma generalista



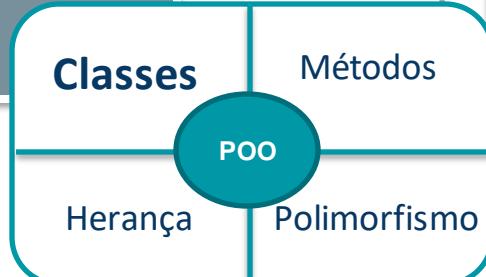
# Overview

Baseado em 4 pilares

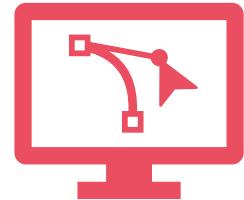
Extrapolar o conceito do problema para um forma generalista


$$f(x,y) := x^2 + y^2$$
$$g(x,y) := x + 20$$

$$C = 2\pi r$$



Decomposição



Design

Reconhecimento de padrões

Abstração

Design de algoritmos

# Overview

Baseado em 4 pilares

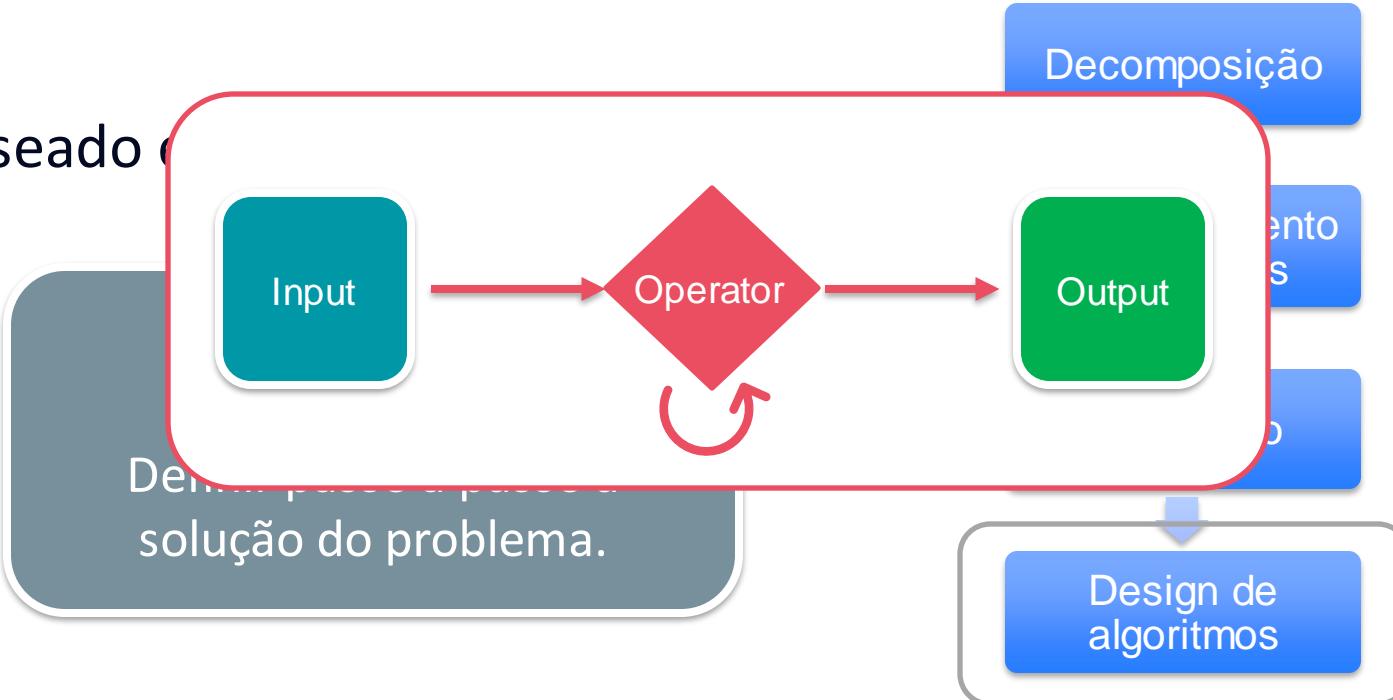
Automatizar

Definir passo a passo a solução do problema.



# Overview

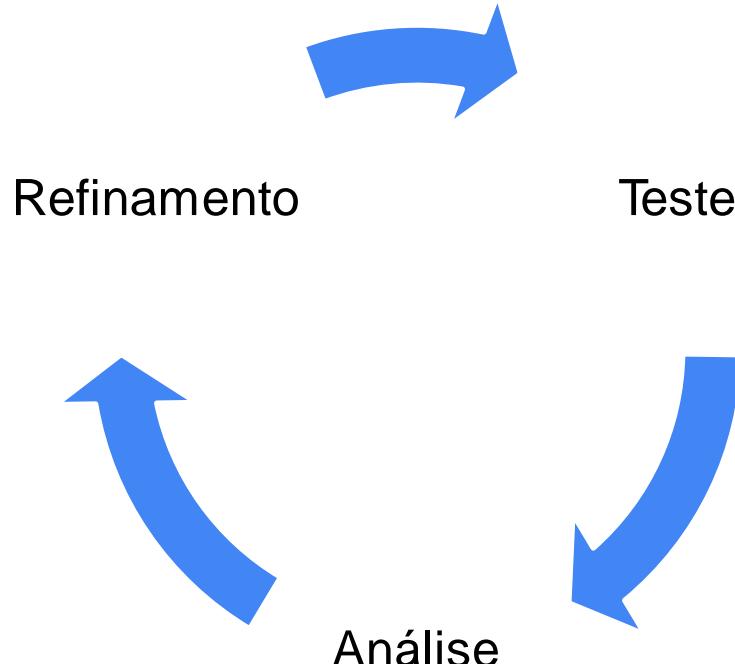
Baseado em



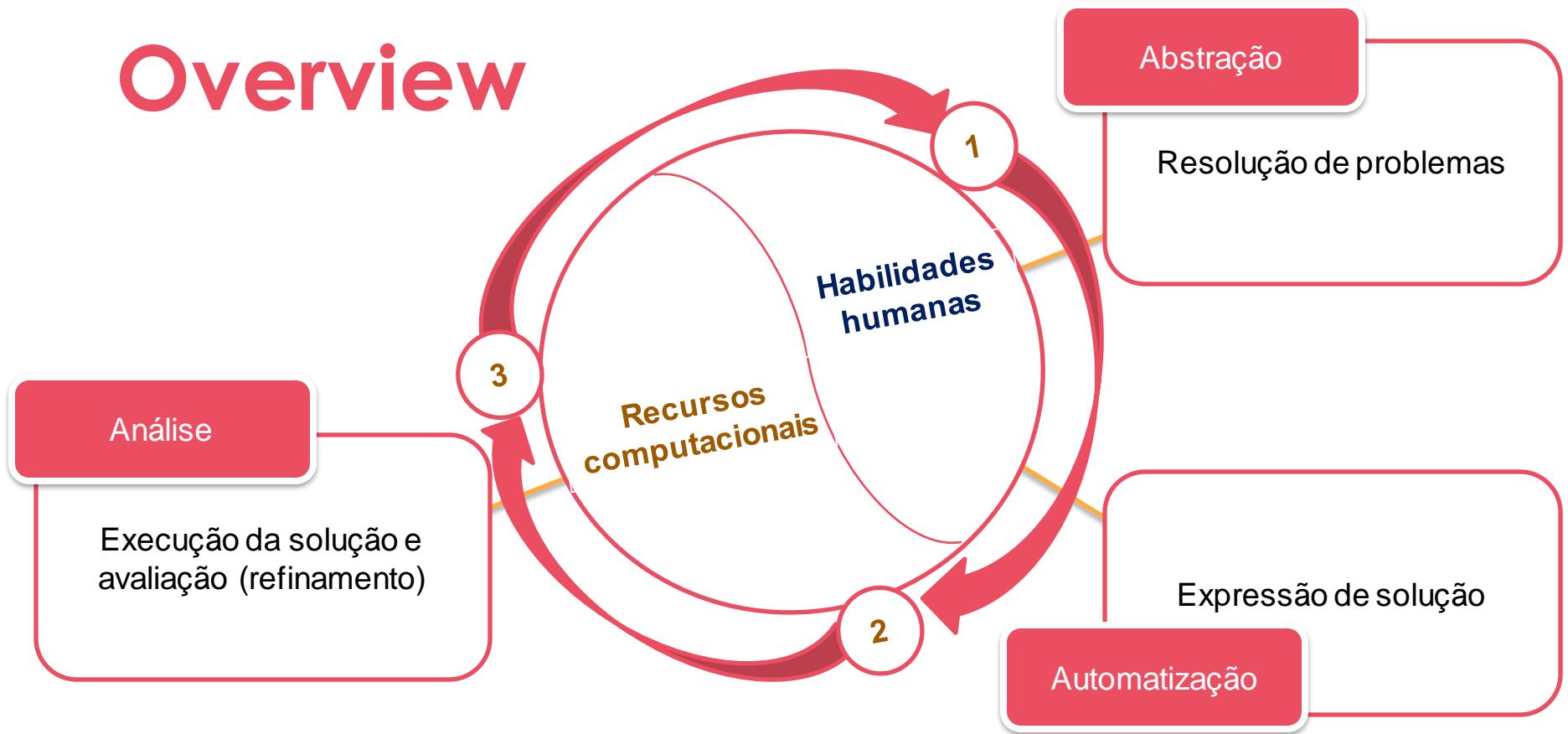
# Overview

## Processo Contínuo

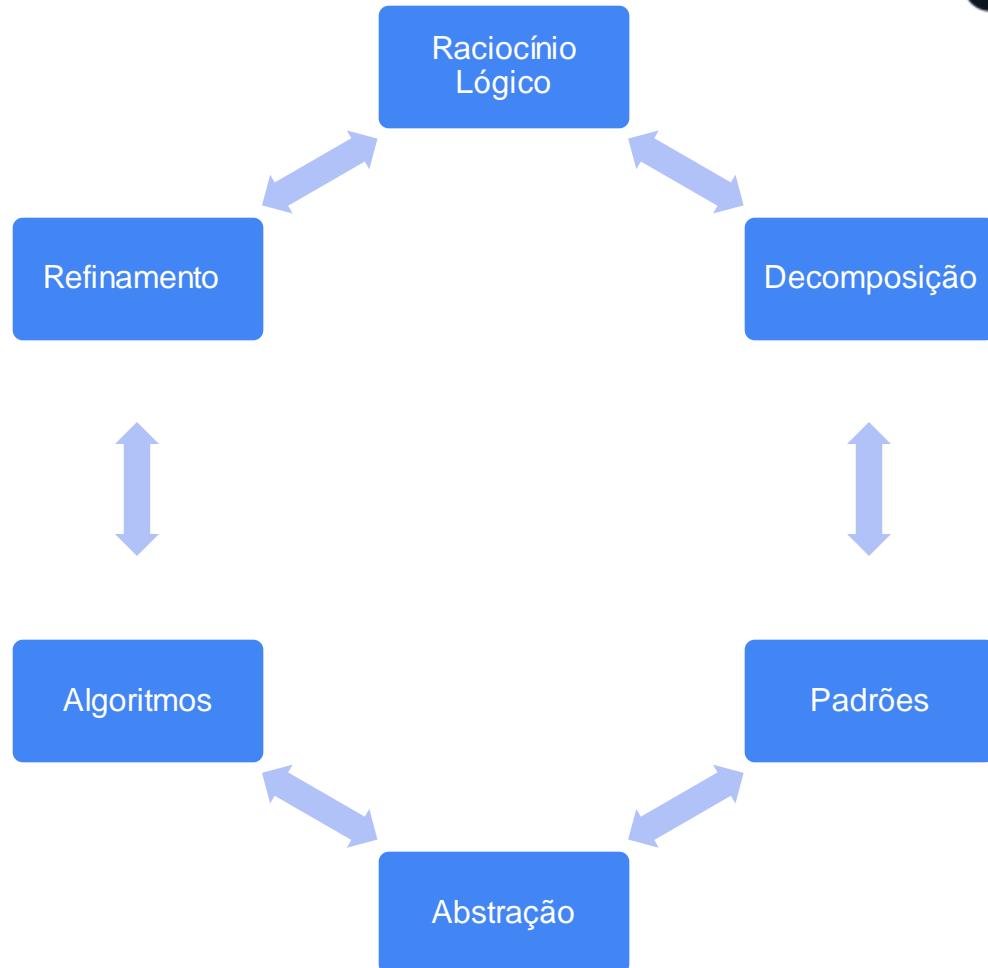
- Definir uma solução
- Testar a solução
- Aperfeiçoamento da solução encontrada



# Overview



# Overview



# Overview

## 1.CS + Math

Desenvolvimento e abstração; Reconhecimento de padrões ...

## 3.CS + Sci/Eng + Math

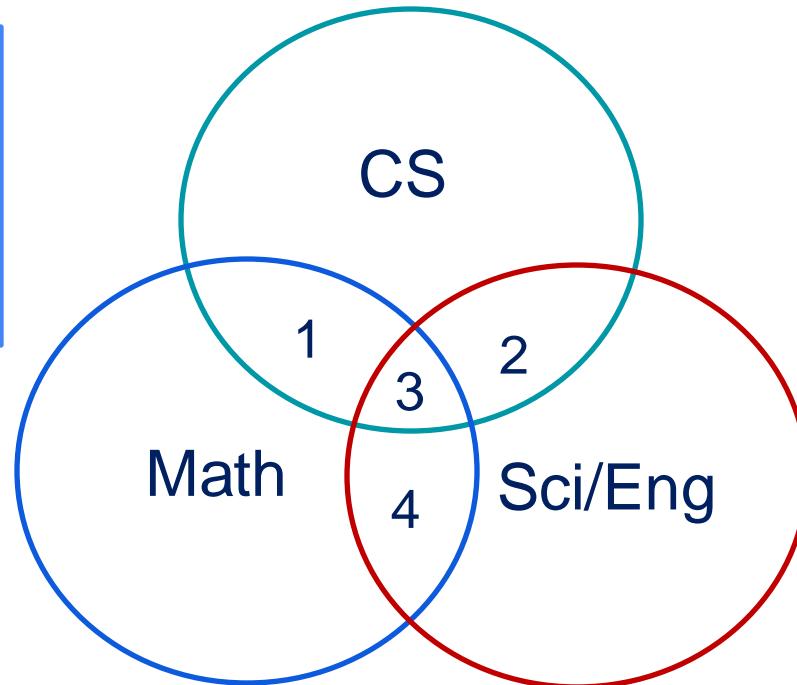
Modelagem; Definição de problemas; Definição e uso de abstrações; Reconhecimento de padrões ...

## 2.CS + Sci/Eng

Análise de dados e design de soluções; Definição e uso de abstrações; Teste e refinamento de algoritmos ...

## 4.Math +Sci/Eng

Desenvolvimento e abstração; Reconhecimento de padrões ...



# Overview

## Exemplos



### Química

- Aperfeiçoamento de reações químicas pela utilização de algoritmos, através da identificação de químicos

### Engenharia

- Simulações de aeronaves executadas via software em detrimento do tunel de vento

# Overview

## Exemplos

### Biologia

- Modelagem e mapeamento do genoma humano

### Computação

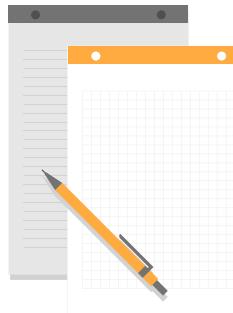
- Simulação de problemas de alta ordem em supercomputadores



# Overview

## Competências

- Pensamento sistemático
- Colaboração dentro da equipe
- Criatividade e design
- Facilitador



## Etapa 2

# Habilidades complementares

// Primeiros passos para começar a  
programar/Pensamento Computacional

# Percorso

**Etapa 1**

Introdução ao pensamento computacional

**Etapa 2**

**Habilidade complementares**

**Etapa 3**

Pilares: Decomposição

# Percurso

**Etapa 4**

Pilares: Padrões

**Etapa 5**

Pilares: Abstração

**Etapa 6**

Pilares: Algoritmos

# Percurso

Etapa 7

Estudo de caso conceitual: perdido

Etapa 8

Estudo de caso aplicado: Soma de um intervalo

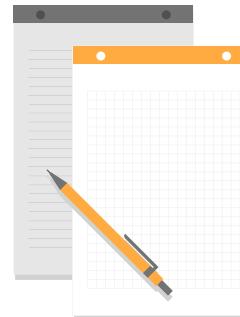
Etapa 9

Estudo de caso aplicado: Adivinhe um número

# Habilidades

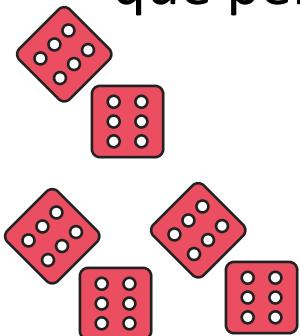
Raciocínio Lógico

Aperfeiçoamento



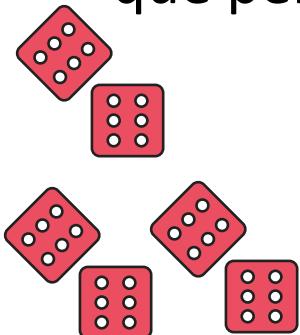
# Raciocínio Lógico

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



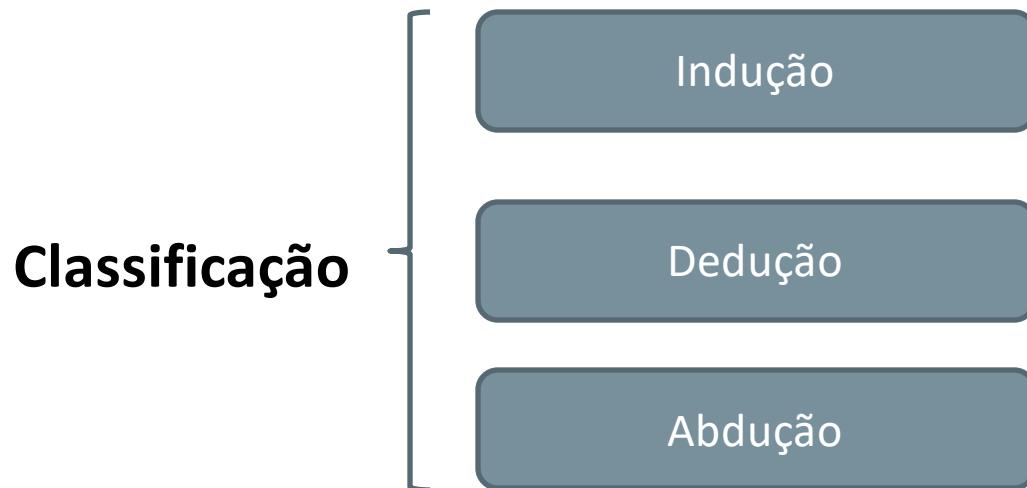
# Raciocínio Lógico

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



Habilidade de treinamento

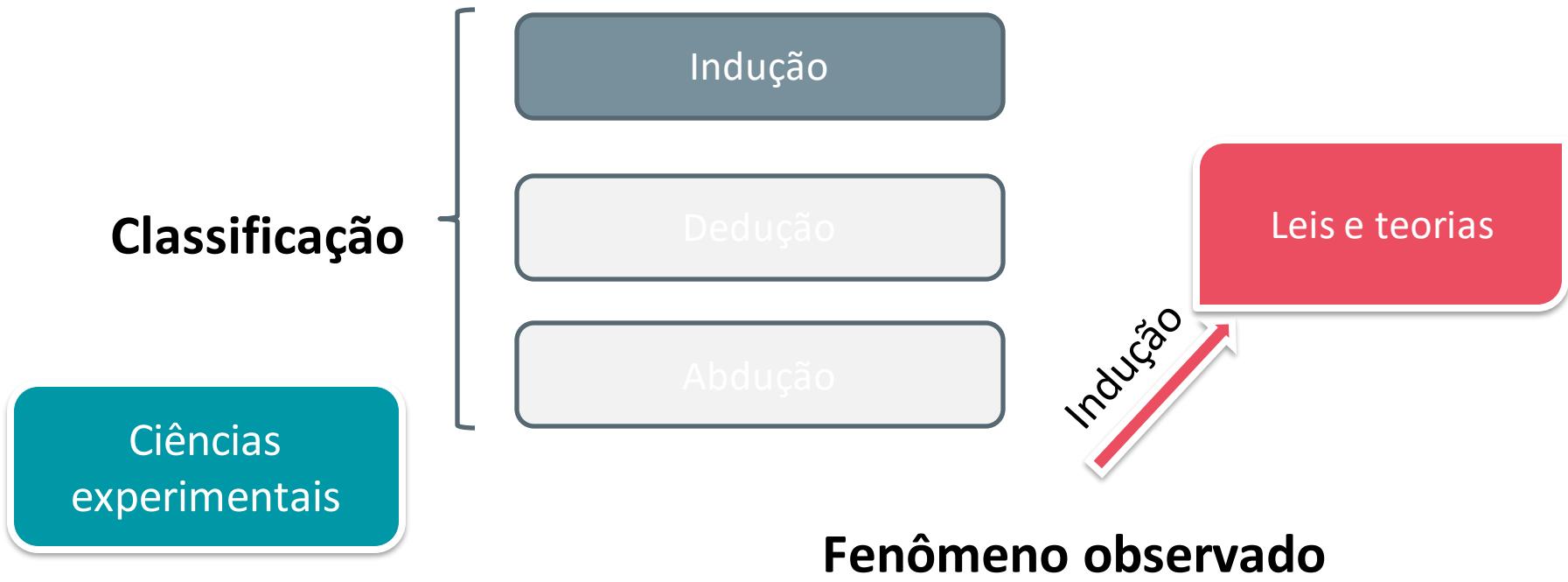
# Raciocínio Lógico



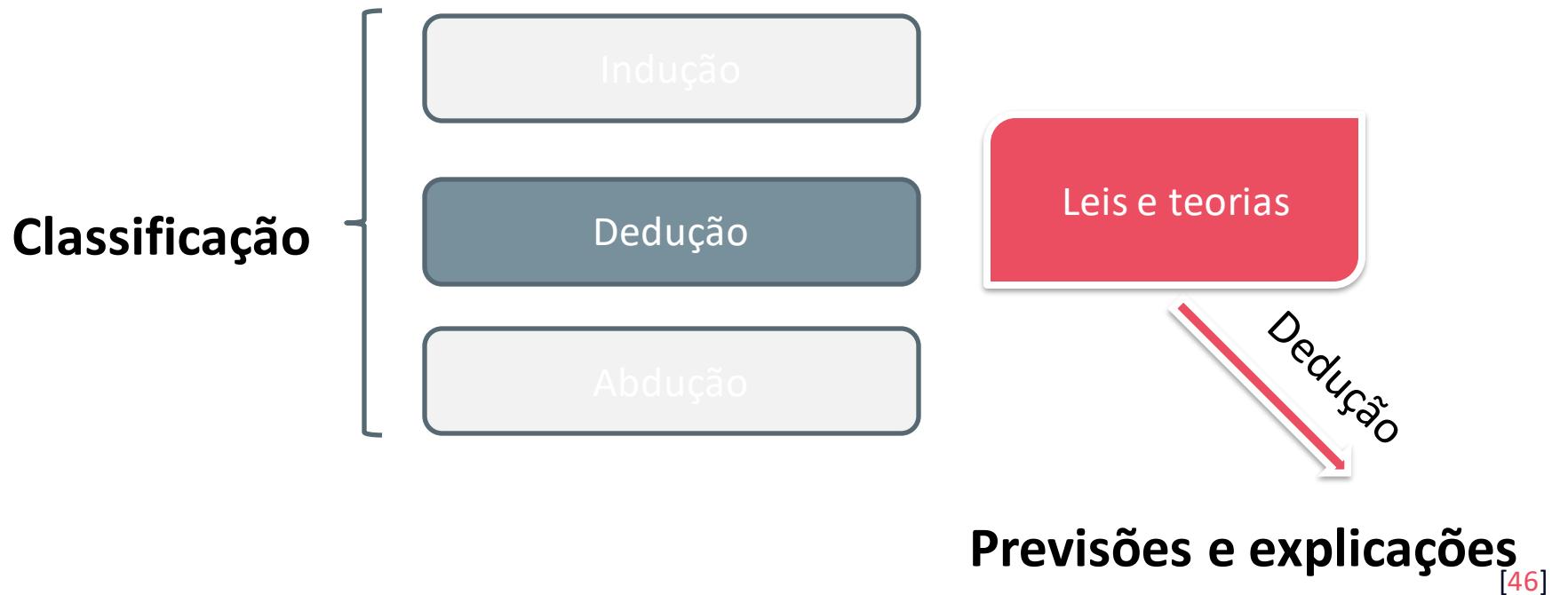
# Raciocínio Lógico



# Raciocínio Lógico



# Raciocínio Lógico



# Raciocínio Lógico

Classificação

Ciências exatas

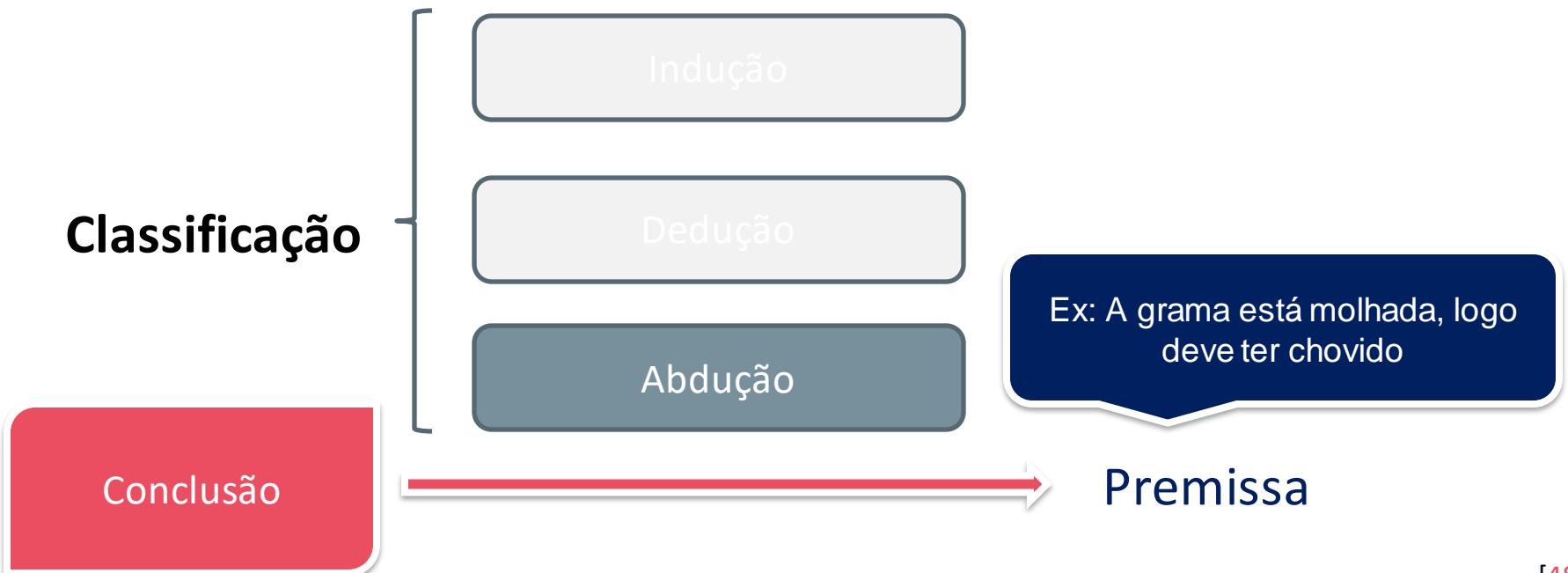


Leis e teorias

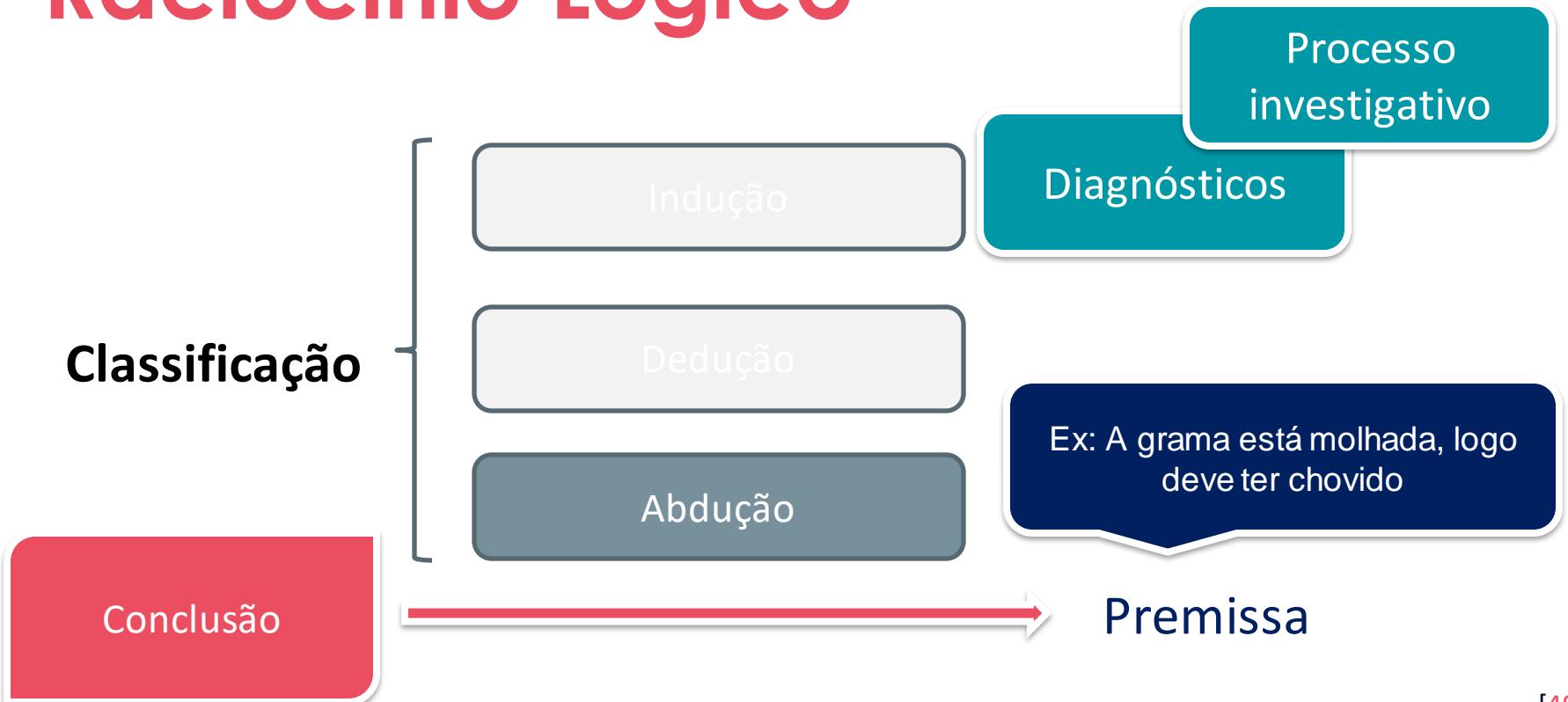
Previsões e explicações

Dedução

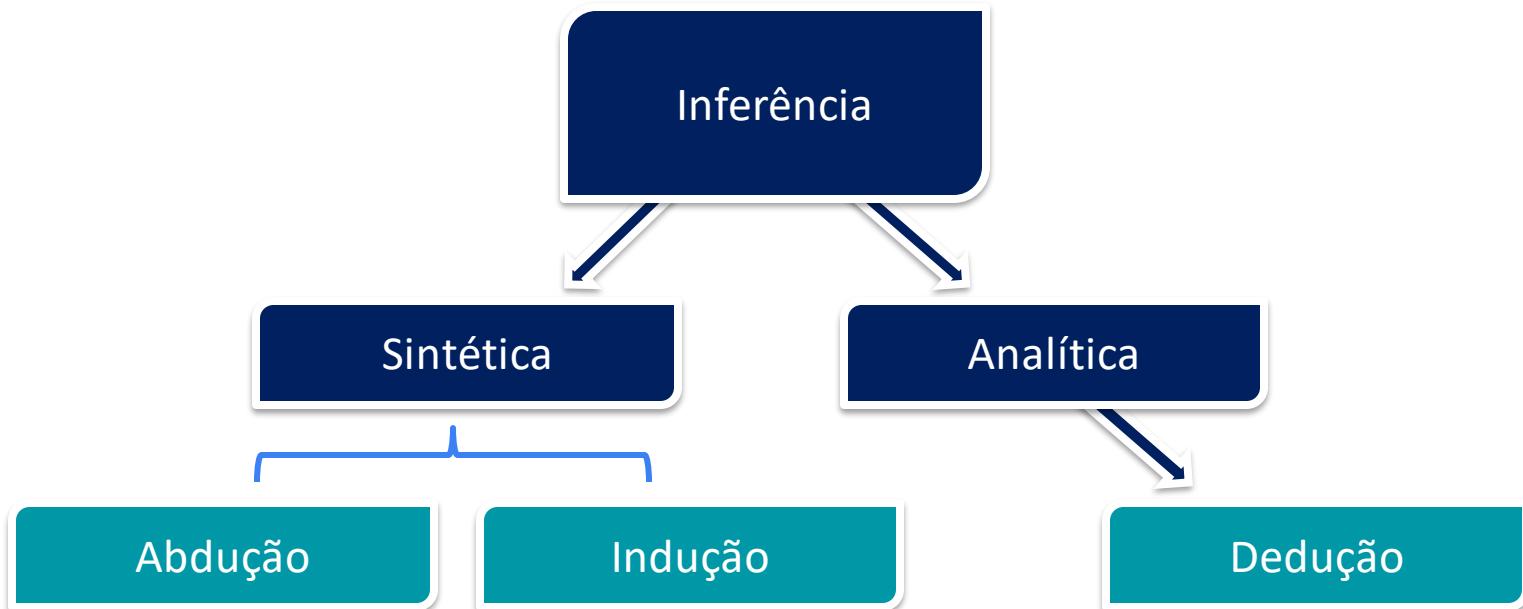
# Raciocínio Lógico



# Raciocínio Lógico

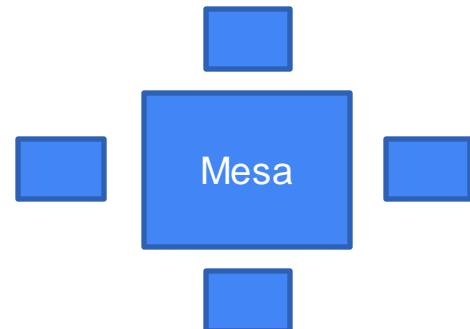


# Raciocínio Lógico



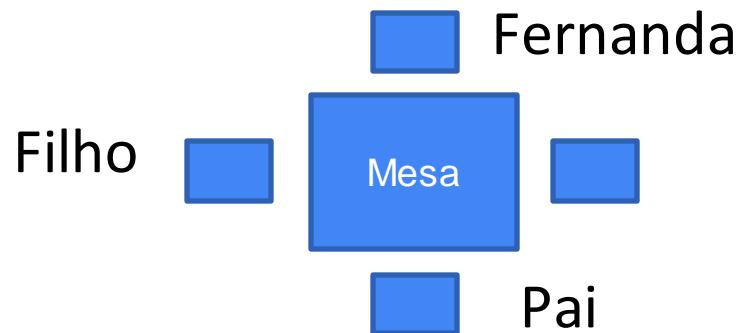
# Raciocínio Lógico

Um pai, mãe e seu casal de filhos estão sentados em uma mesa. Os homens se chamam Roberto e Sérgio, as mulheres Teresa e Fernanda. Sabe-se que o pai está à frente de Fernanda e o filho à esquerda; e que a mãe está do lado direito de Sérgio.



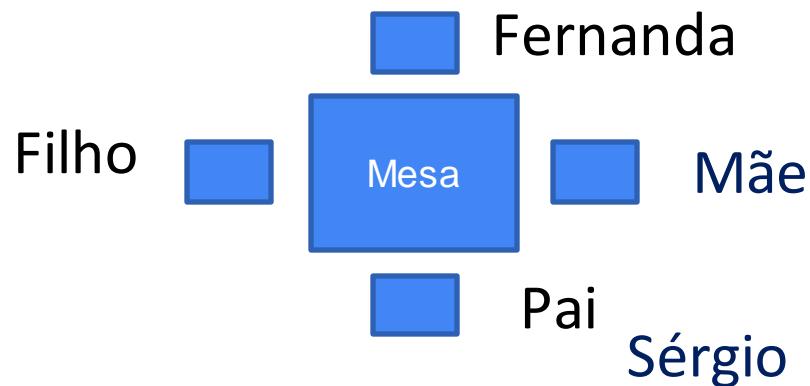
# Raciocínio Lógico

Um pai, mãe e seu casal de filhos estão sentados em uma mesa. Os homens se chamam Roberto e Sérgio, as mulheres Teresa e Fernanda. Sabe-se que o pai está à frente de Fernanda e o filho à esquerda; e que a mãe está do lado direito de Sérgio.



# Raciocínio Lógico

Um pai, mãe e seu casal de filhos estão sentados em uma mesa. Os homens se chamam Roberto e Sérgio, as mulheres Teresa e Fernanda. Sabe-se que o pai está à frente de Fernanda e o filho à esquerda; e que a mãe está do lado direito de Sérgio.



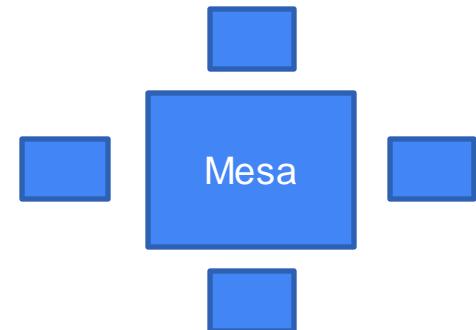
# Raciocínio Lógico

Um pai, mãe e seu casal de filhos estão sentados em uma mesa. Os homens se chamam Roberto e Sérgio, as mulheres Teresa e Fernanda. Sabe-se que o pai está à frente de Fernanda e o filho à esquerda; e que a mãe está do lado direito de Sérgio.

R: Sérgio é o pai

Teresa é a mãe

Fernanda a filha e Roberto é o filho.



# Aperfeiçoamento

Melhoramento

Ato de aperfeiçoar

Aprimoramento

Refinamento

# Aperfeiçoamento

Melhoramento

Ato de aperfeiçoar

Aprimoramento

A partir de uma solução,  
determinar pontos de melhora e  
refinamento

Refinamento

# Aperfeiçoamento

Ato de aperfeiçoar

- Encontrar solução eficiente
  - Otimizar processos
  - Simplificar linhas de códigos
  - Funções bem definidas
- 
- Melhor uso de recursos
- Melhorar códigos e algoritmos

## Etapa 3

# Pilares: Decomposição

// Primeiros passos para começar a  
programar/ Pensamento Computacional/

# Percorso

**Etapa 1**

Introdução ao pensamento computacional

**Etapa 2**

Habilidade complementares

**Etapa 3**

**Pilares: Decomposição**

# Percurso

**Etapa 4**

Pilares: Padrões

**Etapa 5**

Pilares: Abstração

**Etapa 6**

Pilares: Algoritmos

# Percurso

Etapa 7

Estudo de caso conceitual: perdido

Etapa 8

Estudo de caso aplicado: Soma de um intervalo

Etapa 9

Estudo de caso aplicado: Adivinhe um número

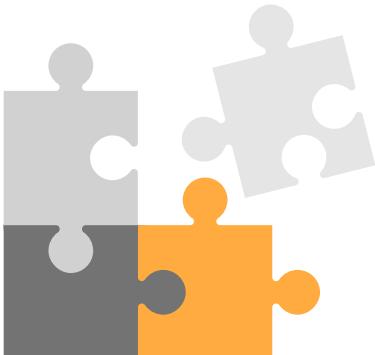
# Objetivo Geral

É preciso coompreender como executar cada etapa de um pensamento computacional.

Dessa forma, as aulas subsequentes são dedicadas à compreensão individual de cada pilar desse processo de pensamento.

# Decomposição

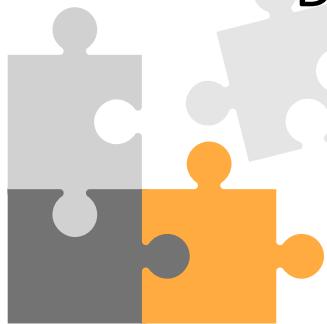
"If you can't solve a problem,  
then there is an easier  
problem that you can solve: find it"



George Polya – professor e matemático

# 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."

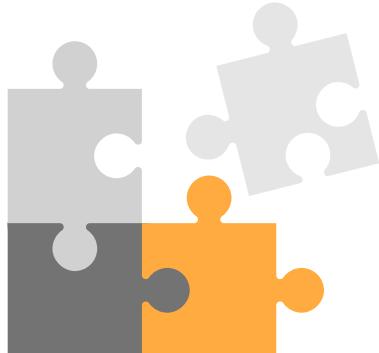
# Decomposição

Estratégia

Processo de quebrar e determinar  
partes menores e gerenciáveis



Análise

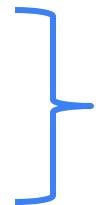


# Decomposição

Estratégia

Estudar, explorar

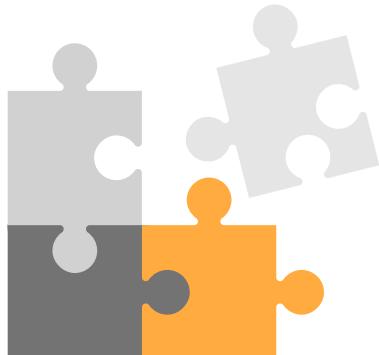
Processo de quebrar e determinar  
partes menores e gerenciáveis



Análise

"realizar exame  
detalhado"

Decompor em elementos  
constituíntes



# Decomposição

Estratégia

Combinar os elementos  
recompondo o problema original

} Síntese



# Decomposição

Estratégia

Processo de  
reconstrução

Combinar os elementos  
recompondo o problema original

Síntese

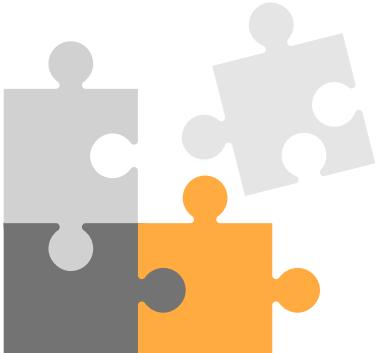


Consiste em reunir  
elementos distintos  
em um único  
elemento

Fundir os  
elementos de  
maneira coerente

# Decomposição

Ordem de execução de tarefas menores



Estratégia

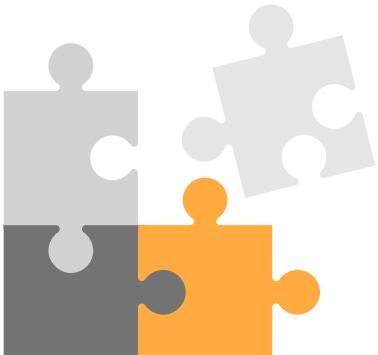


Sequêncial

Paralelo

# Decomposição

Ordem de execução de tarefas menores



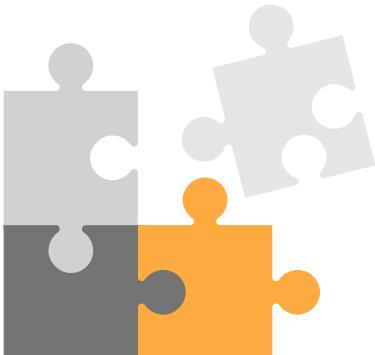
Estratégia

Sequêncial

Dependência entre tarefas.  
Executadas em "fila"

# Decomposição

Ordem de execução de tarefas menores



Estratégia

Paralelo

Tarefas podem ser executadas concomitantemente.



Eficiência



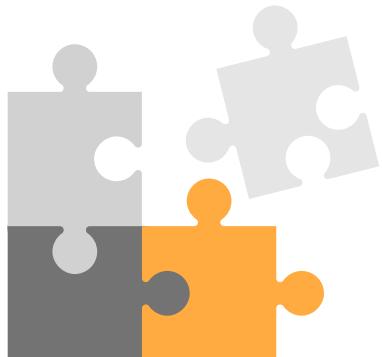
Tempo

# Decomposição

Pequenos  
problemas

Variáveis

Segmentação



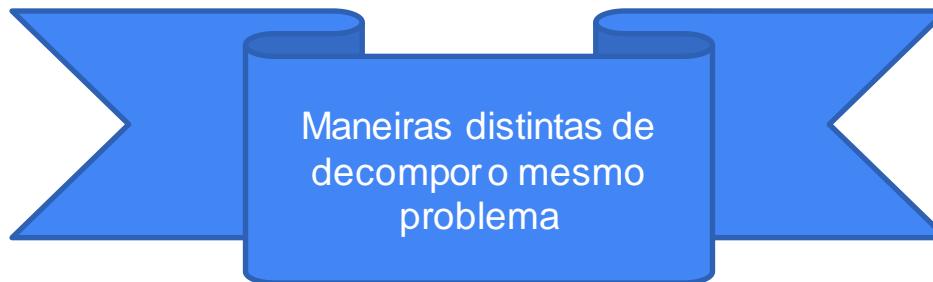
# Decomposição

- Não basta aplicar
- Desenvolver a decomposição "by yourself"

Pequenos  
problemas

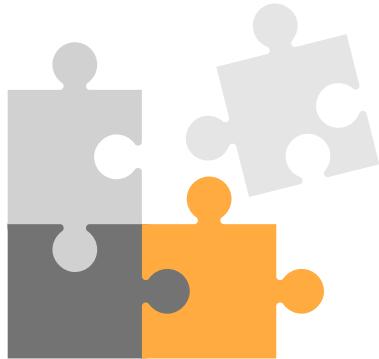
Variáveis

Segmentação



# Decomposição

Como decompor?

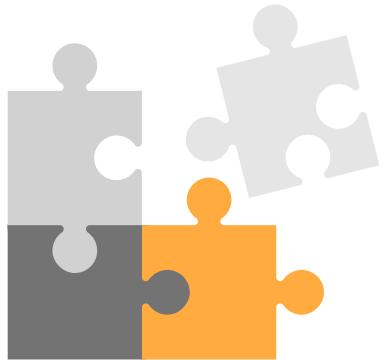


Decomposição

# Decomposição



Exemplos



# Decomposição

Ex do cotidiano: cozinhar

- Identificar os ingredientes
- Determinar as etapas (sequêncial ou paralelo)
- Executar cada etapa
- Agregar os ingredientes para finalizar (Recompor com coerência)



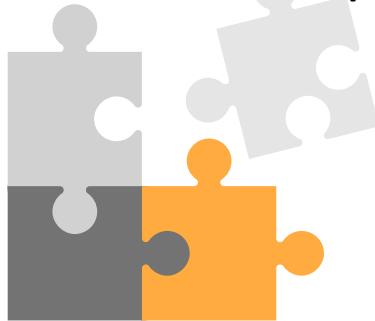
# Decomposição

Ex do cotidiano: funcionamento de uma bike

- Identificar os componentes
- Papel de cada componente
- Interdependência das peças



Funcionamento  
do sistema



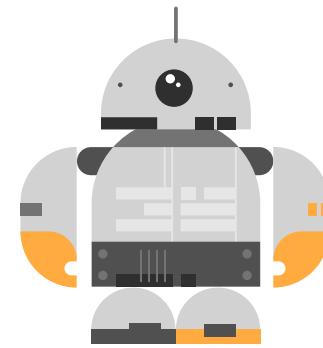
# Decomposição

Exemplo: criar um app

- Finalidade
- Interface
- Funcionalidades
- Pré-requisitos



Definição de componentes e etapas



Desenvolvimento mais eficiente

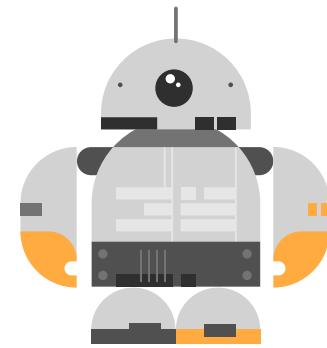
# Decomposição

Exemplo: criar um app

- Finalidade
- Interface
- Funcionalidades
- Pré-requisitos



Definição de  
componentes e etapas



Desenvolvimento mais  
eficiente

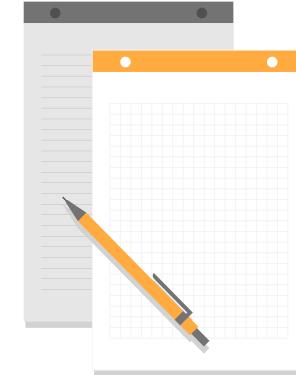
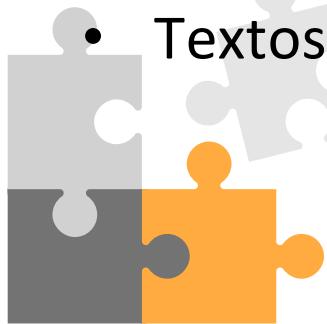
# Decomposição

Exemplo: Artigo

- O que será abordado?
- Estrutura
- Conteúdo de cada tópico
- Textos de conexão



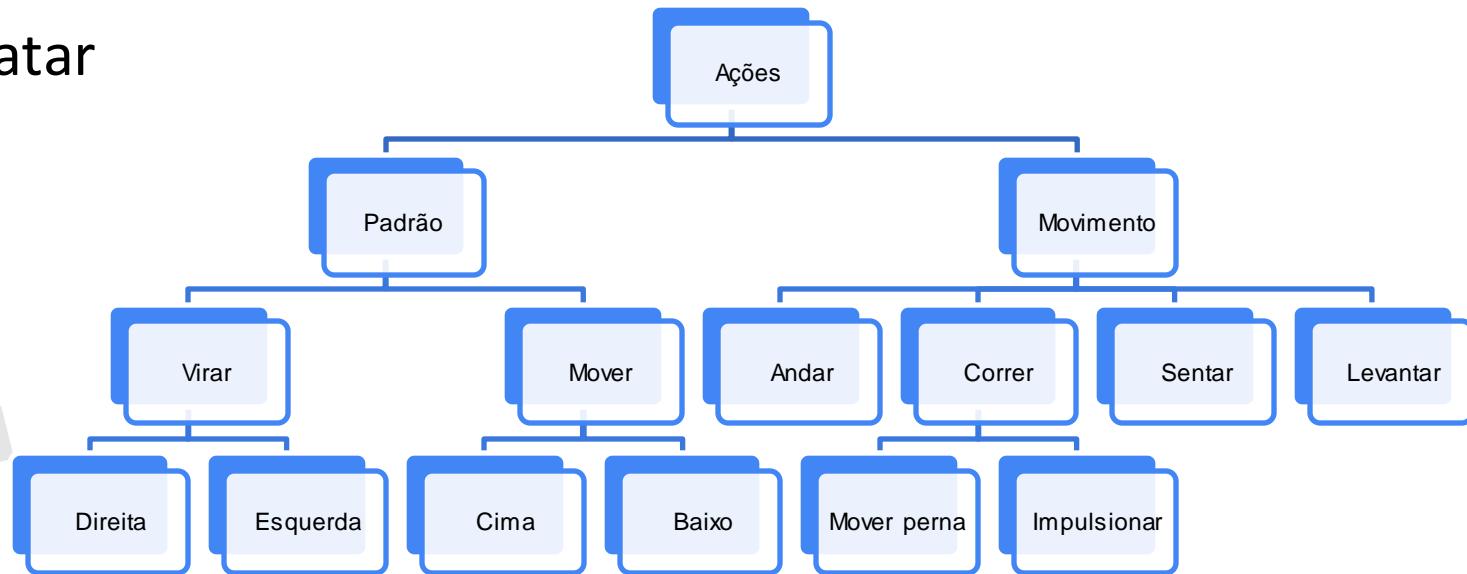
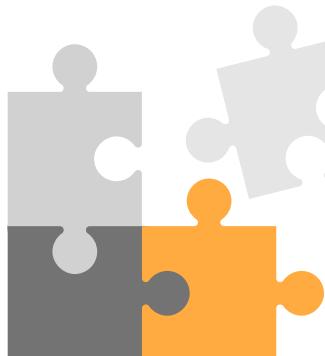
Definição de  
componentes e etapas



Sequêncial ou paralelo  
Em ordem ou não

# Decomposição

Exemplo: movimentos  
de um avatar



## Etapa 4

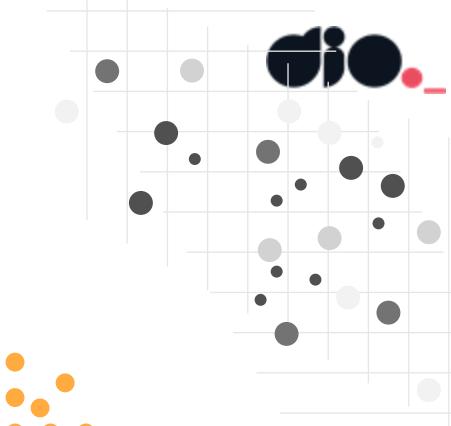
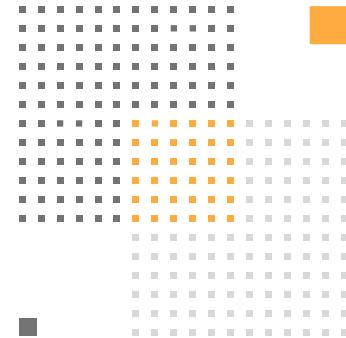
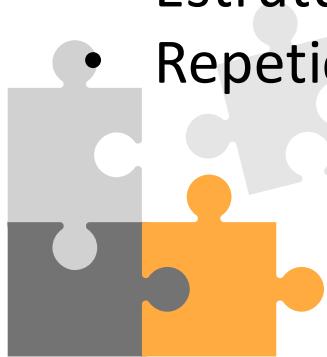
# Pilares: Padrões

// Primeiros passos para começar a  
programar/ Pensamento Computacional/

# Padrões

Reconhecimento de Padrões

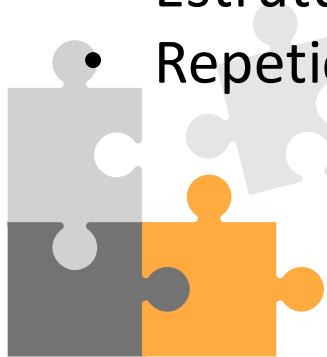
- Modelo base
- Estrutura invariante
- Repetição



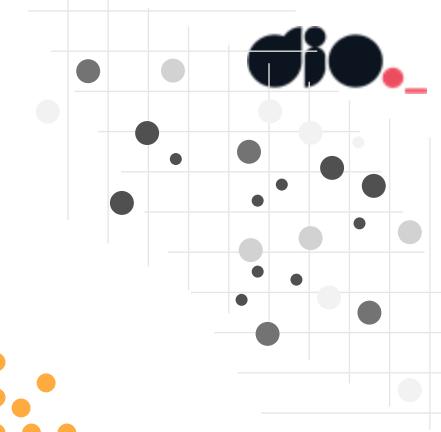
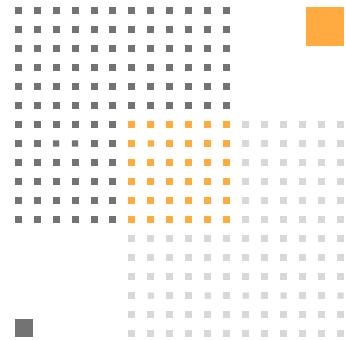
# Padrões

## Reconhecimento de Padrões

- Modelo base
- Estrutura invariante
- Repetição



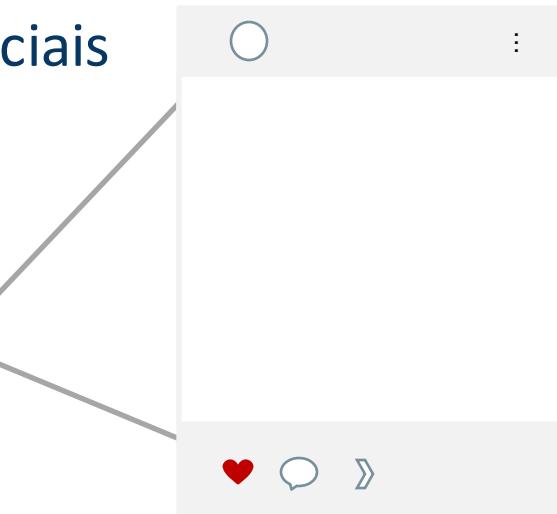
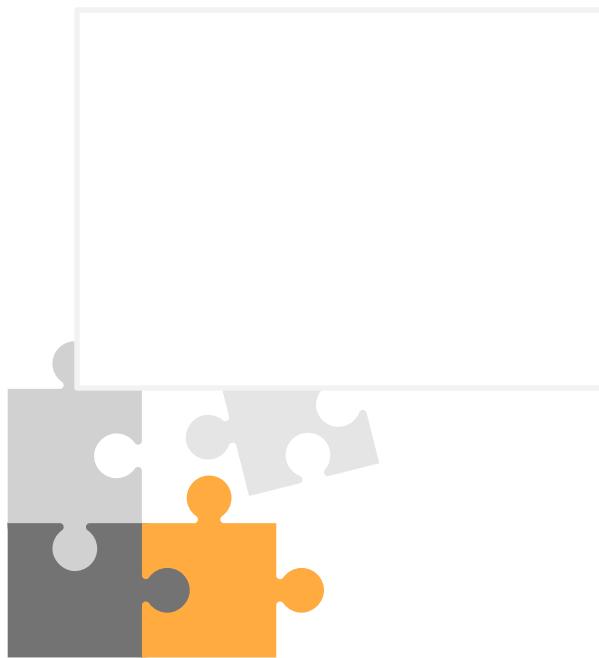
Similaridades e  
diferença



Para se enviar varias fotos ou posts para as redes socias utiliza um processo de compressão.

# Padrões

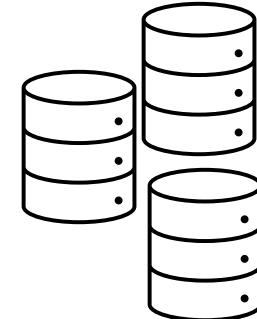
Ex: fotos de redes sociais



Compressão



Salvar no servidor  
de dados



# Padrões

Ex: fotos de redes sociais



Processo utilizado por  
plataformas diferentes

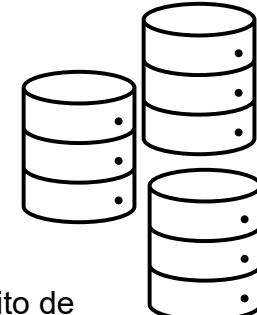
Este processo é um padrão para todas as plataformas, pode ser feito de formas diferentes.



Compressão

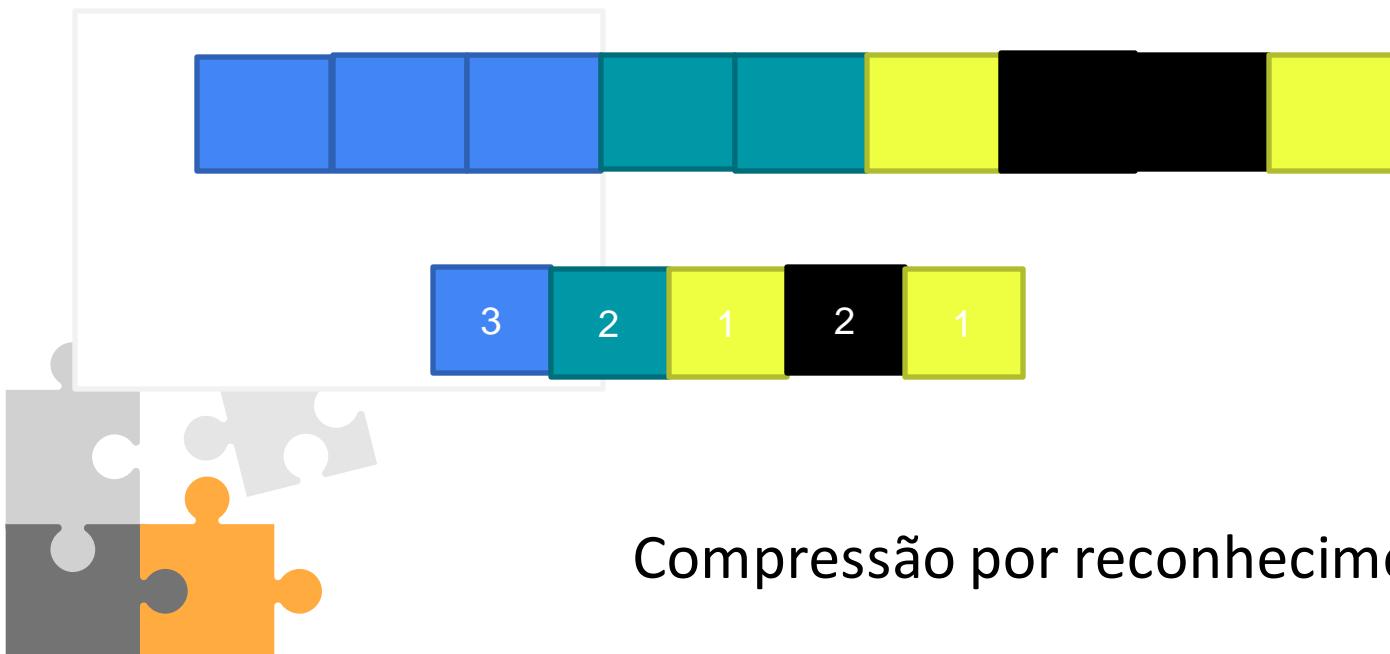


Salvar no servidor  
de dados



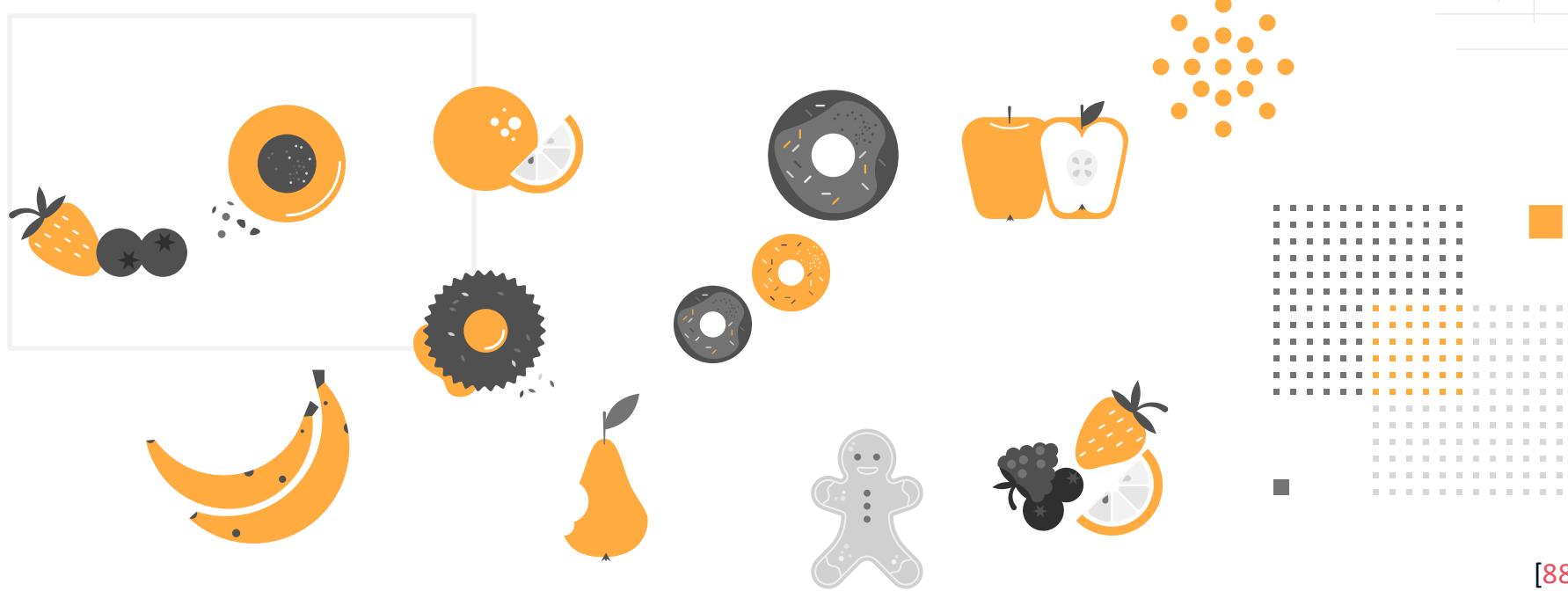
# Padrões

Ex: Compressão de dados



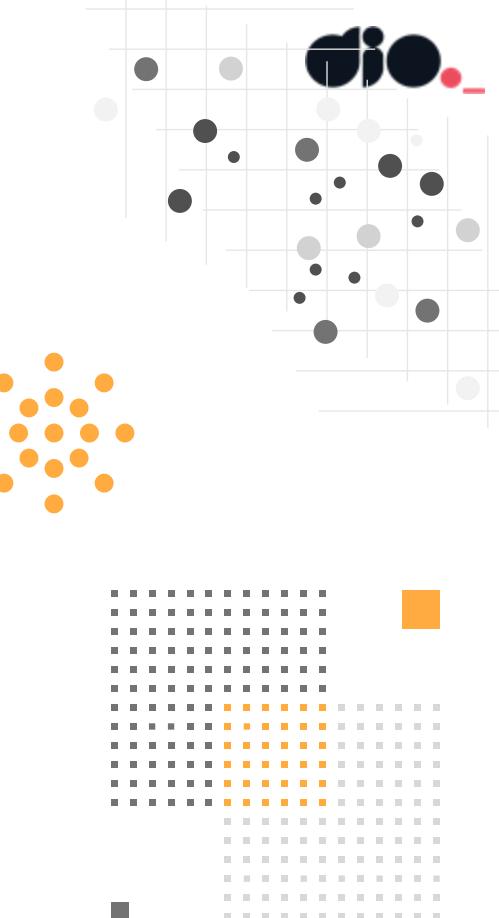
# Padrões

Seres vivos x padrões



# Padrões

Seres vivos x padrões



Os seres vivos faz o reconhecimento de padrões de forma intuitiva, levando em conta as características, e a similaridades dos elementos.

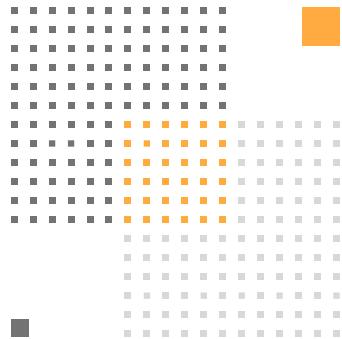


# Padrões

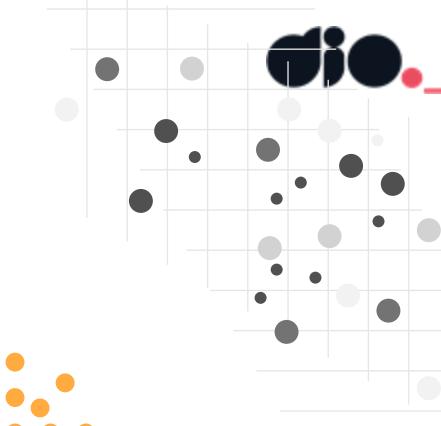
Reconhecimento de Padrões



- Por que determinar padrões?



# Padrões

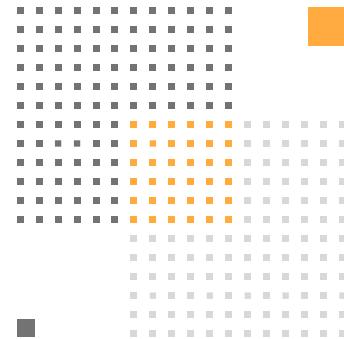


Reconhecimento de Padrões



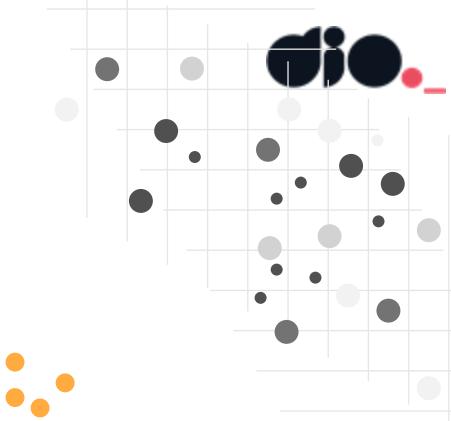
- Por que determinar padrões?

Generalizar, com objetivo de obter  
resolução para problemas diferentes



Generalizar: tornar(-se) geral; estender(-se); propagar(-se); universalizar(-se);  
tornar(-se) geral; estender(-se); propagar(-se); universalizar(-se).

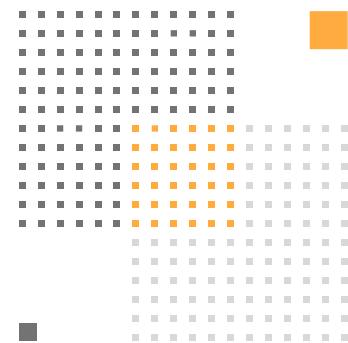
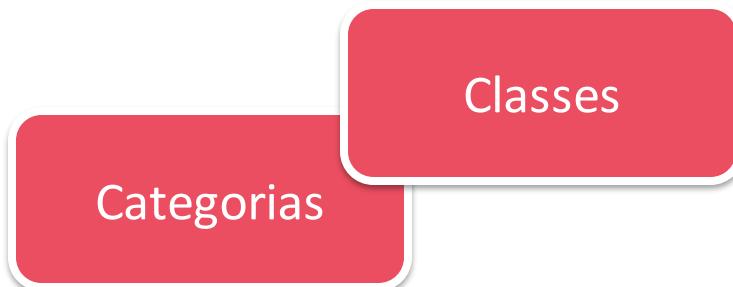
# Padrões



Reconhecimento de Padrões



- Por que determinar padrões?



Os padrões são reconhecidos através de classes e categorias, onde as categorias depende do tipo de mídia e do domínio, para classificar os objetos e o contexto.



# Padrões

## Reconhecimento de Padrões

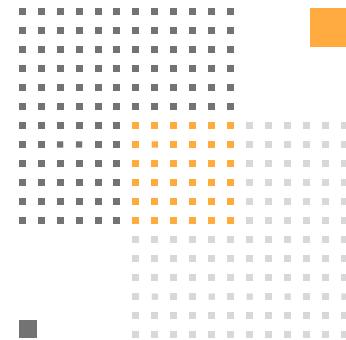


- Por que Tipo de media drões?

Dependem do domínio

Categorias

Classes



# Padrões

Através de:

- Grau de similaridade
- Grupos conhecidos x objeto desconhecido

Como são reconhecidos

Pelo ser humano



# Padrões

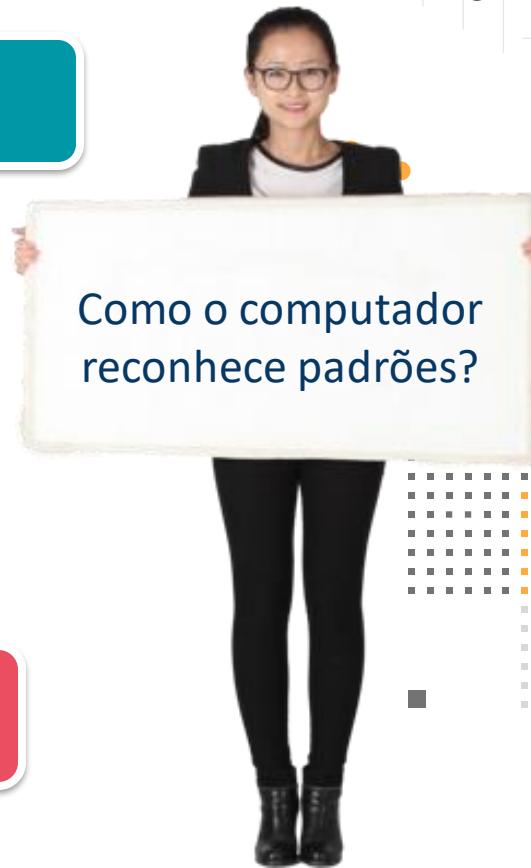
Pelo ser humano

- Grau de similaridade
- Grupos conhecidos x objeto desconhecido

O computador reconhece os padrões por:

Obs: O computador se não tiver a informação registrada não consegue fazer a comparação, pois não usa a objetividade como um ser humano.

COMPARAÇÃO



Como o comutador é objetivo e direto para fazer o reconhecimento de padrões temos que:

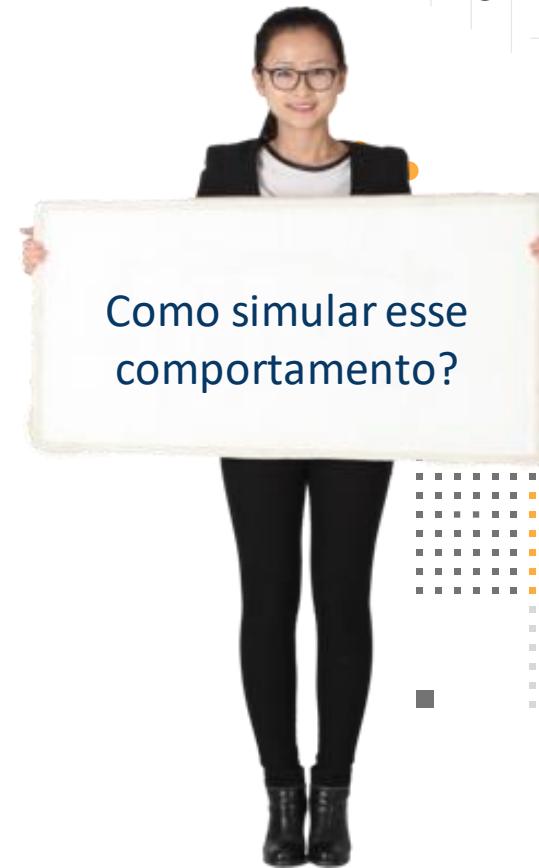
# Padrões

Fazer uma

Representação de atributos

Para que o computador possa ter um

Aprendizado – conceito  
associado ao objeto



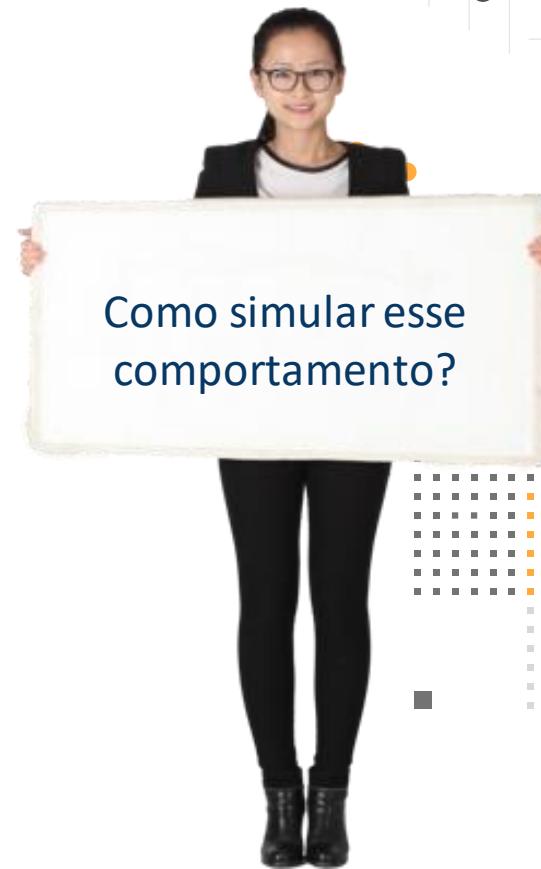
# Padrões

Representação de atributos

Aprendizado – conceito  
associado ao objeto

Para poder

Armazenar dados



# Padrões

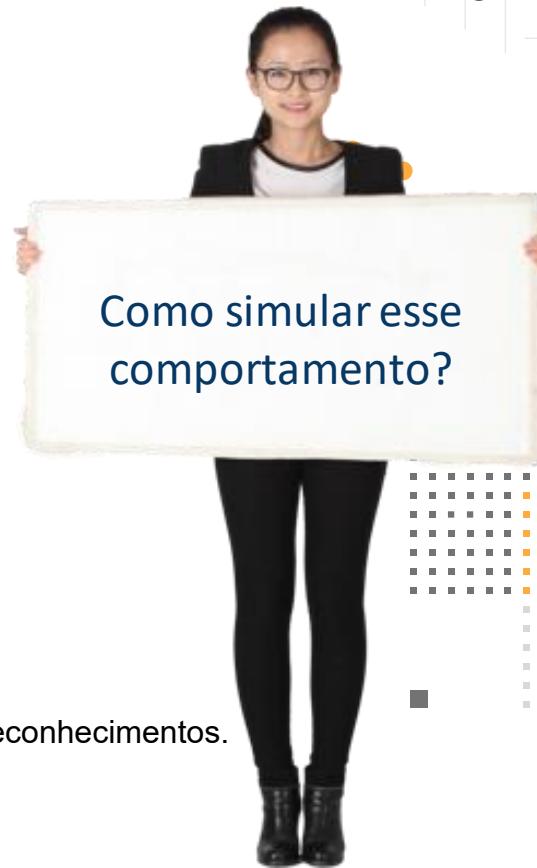
Representação de atributos

Aprendizado – conceito associado ao objeto

Armazenar dados

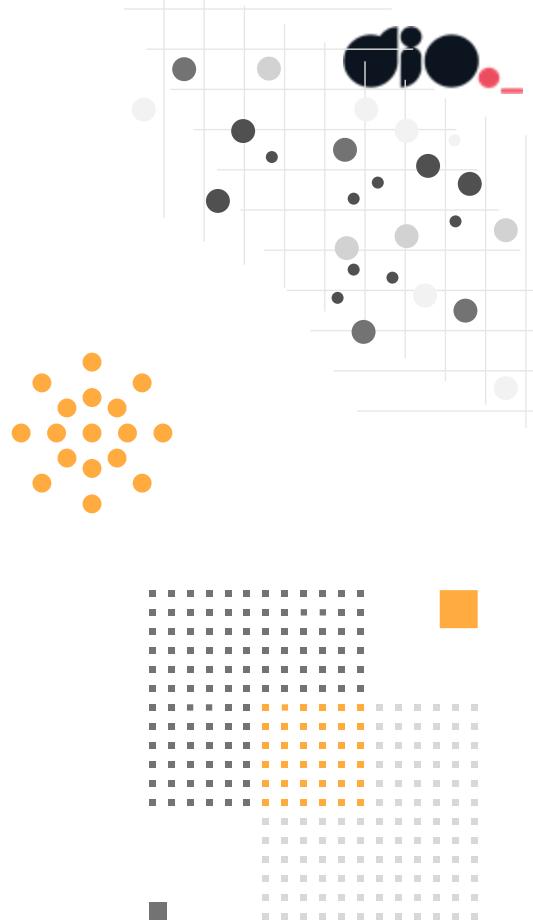
Regras de decisão

Para outris reconhecimentos.



# Padrões

Em resumo ...



A deteção de:

# Padrões

vem da ideia de:

Extração de características

Para ter uma:

Classificação de dados



Abordagem

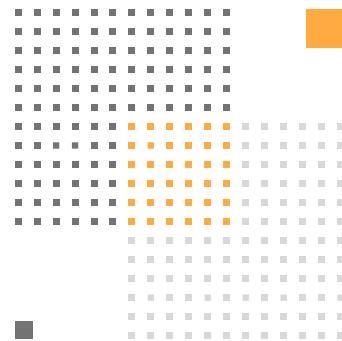
É usado em:

DIFERENTES

Métodos

Aplicações

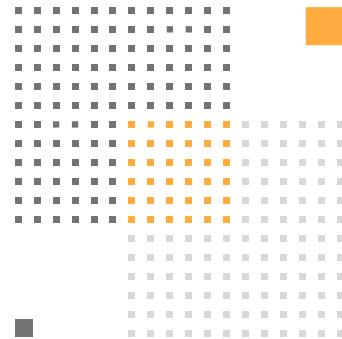
usada no reconhecimento de padrões.



# Padrões

Aplicações

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



# Padrões

Machine Learning

Aplicações

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

Redes Neurais



Inteligência Artificial

Ciência de dados

## Etapa 5

# Pilares: Abstração

// Primeiros passos para começar a  
programar/ Pensamento Computacional/

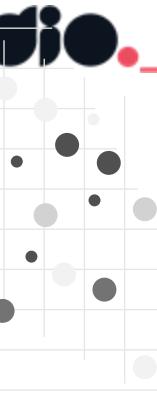
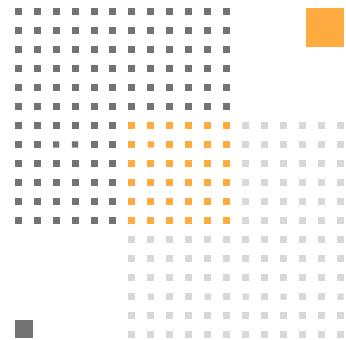
# Abstração

Generalização /  
Abstração

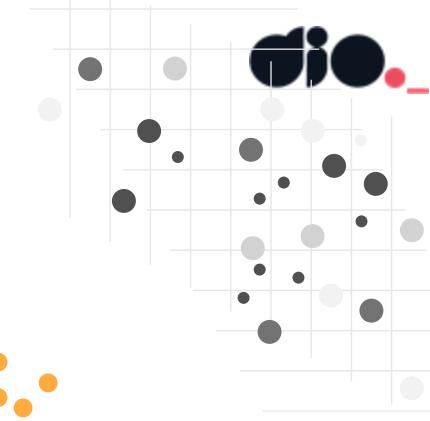
Digite o texto aqui

## ABSTRAIR

Observar, um ou mais elementos,  
avaliando características e  
propriedades em separado



# Abstração



Generalização /  
Abstração

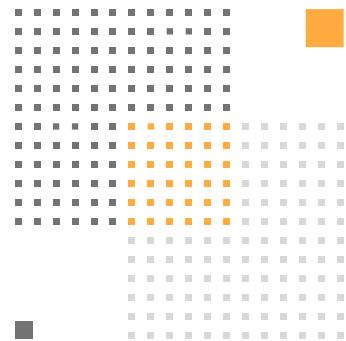


## ABSTRAÇÃO

Processo intelectual de  
isolamento de um objeto da  
realidade

## ABSTRAIR

Observar, um ou mais elementos,  
avaliando características e  
propriedades em separado



# Abstração

Abstrair é detectar características.

## ABSTRAIR

Observar, um ou mais elementos, avaliando características e propriedades em separado

## Generalização / Abstração

## ABSTRAÇÃO

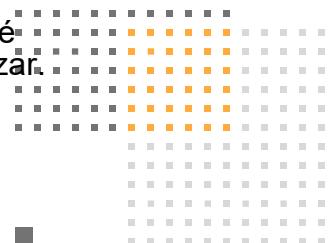
Processo intelectual de isolamento de um objeto da realidade

Abstração e extrapolar um objeto do mundo real para o mundo das ideias.

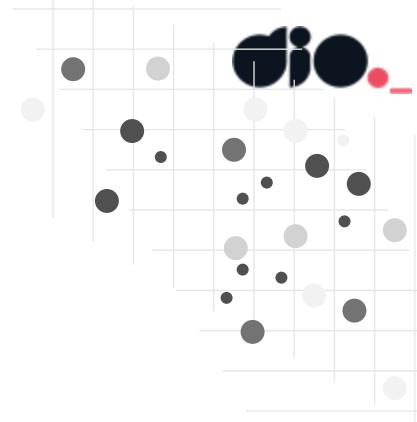
## Generalizar

Tornar-se geral, mais amplo, extensão

Abstrair é generalizar.

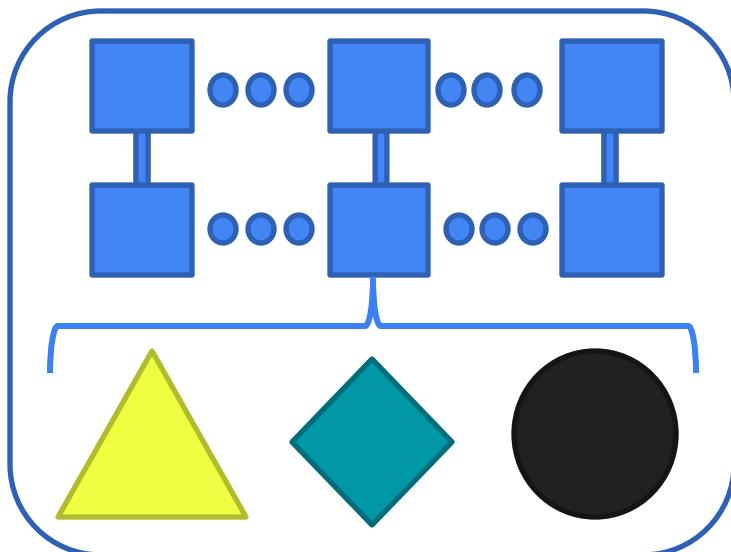


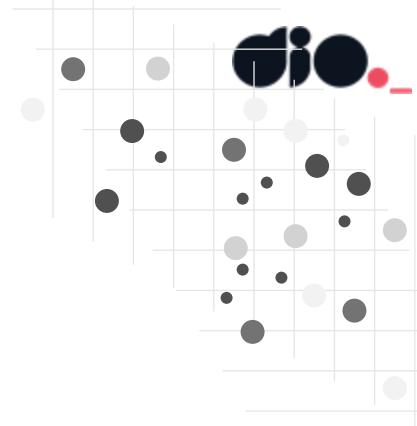
# Abstração



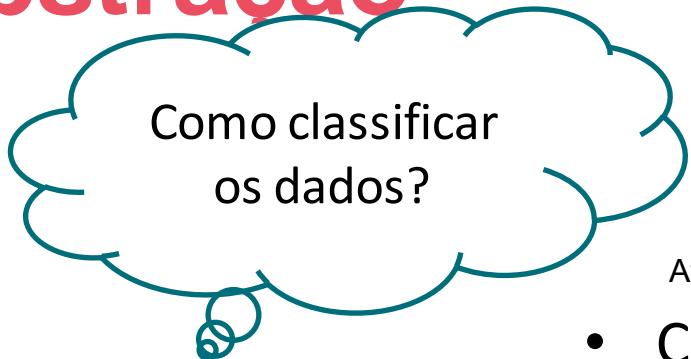
## Generalização / Abstração

**Generalização**, na lógica, é a operação intelectual que consiste em reunir numa classe geral, um conjunto de seres ou fenômenos similares.





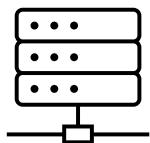
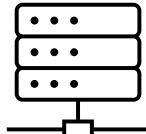
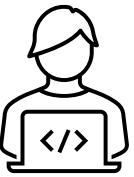
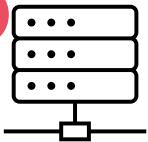
# Abstração



Através de:

- Características
- Pontos essenciais
- Generalizar x detalhar

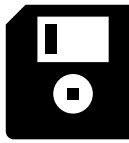
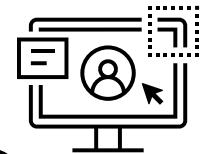
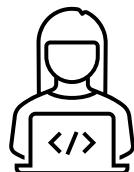
# Abstração



REPRESENTAÇÃO



1010 1010 1010  
1010 1010 1010



# Abstração



Representação de dados

Estudantes



# Abstração



Estudantes

Representação de dados

CARACTERÍSTICAS

# Abstração

## Representação de dados



Estudantes

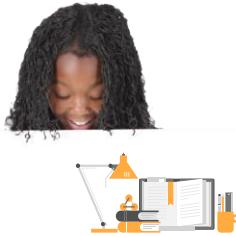


### CARACTERÍSTICAS

- Nome completo
- Matrícula
- Endereço
- Campus
- Curso
- Telefone
- Email
- ...
- Trabalho
- Tem filhos
- Programa preferido
- Livro preferido
- Tamanho do calçado
- ...

Os pontos essenciais são levados em conta e os detalhes são deixados de lado.

# Abstração



Estudantes

Pontos  
essenciais

esentação de dados

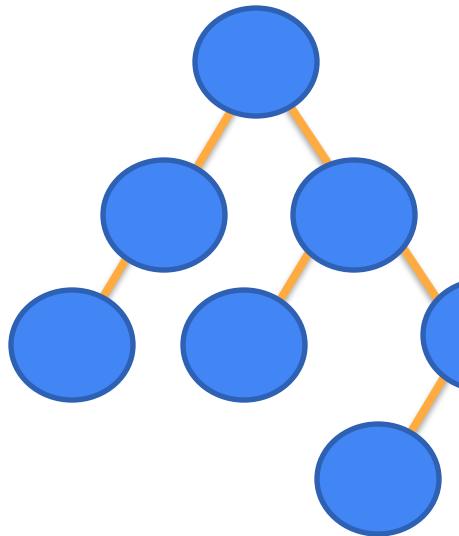
CARACTERÍSTICAS

- Nome completo
- Matrícula
- Endereço
- Campus
- Curso
- Telefone
- Email
- ...

Detalhes

- Trabalho
- Tem carro
- Faz exercícios
- Lembra de tudo
- Tá sempre com calçado
- ...

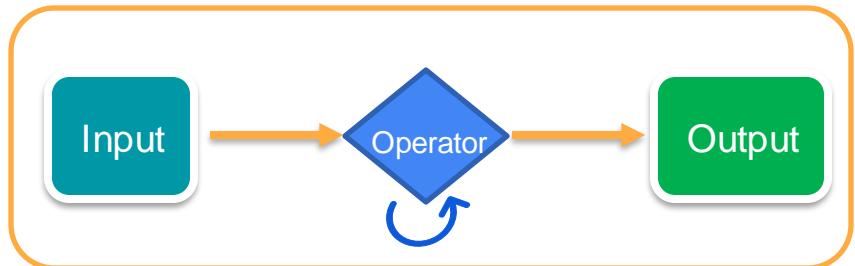
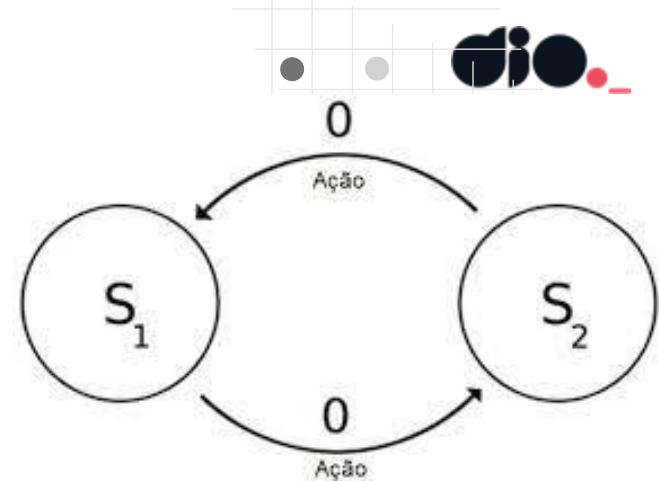
# Abstração



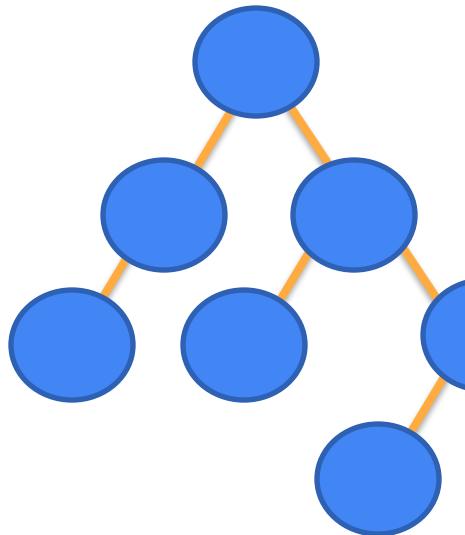
python™



Conceitos baseados  
em abstrações



# Abstração

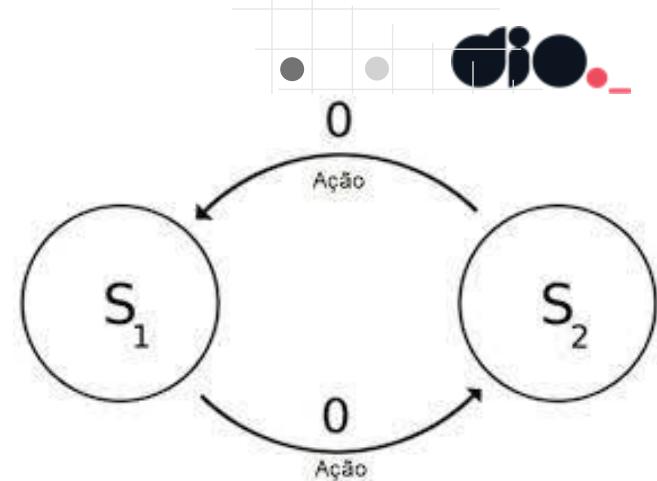


python™



Conceitos baseados  
em abstrações

algoritmo de pesquisa  
e ordenação.

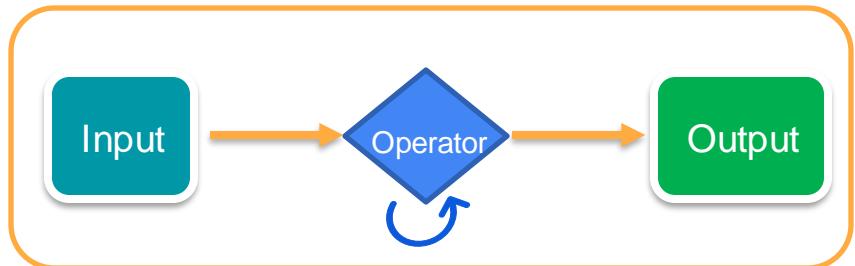
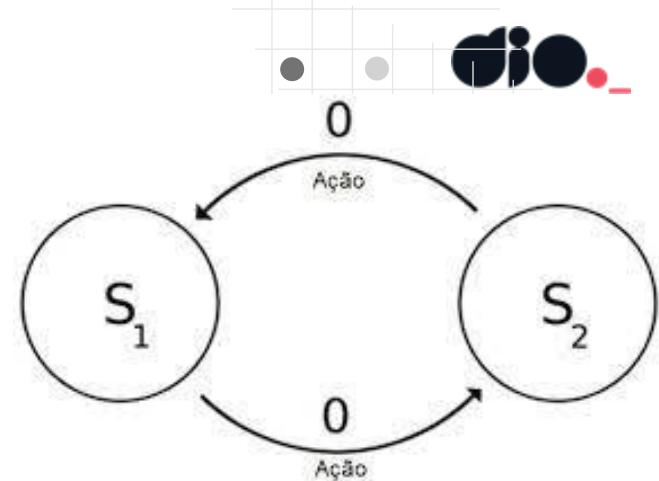


# Abstração

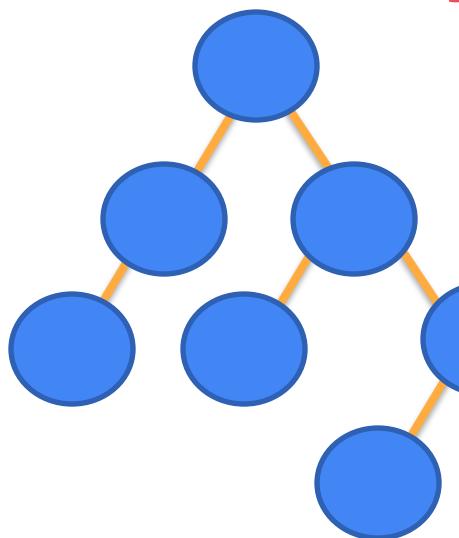
Tipo abstrato  
de dados.



Conceitos baseados  
em abstrações



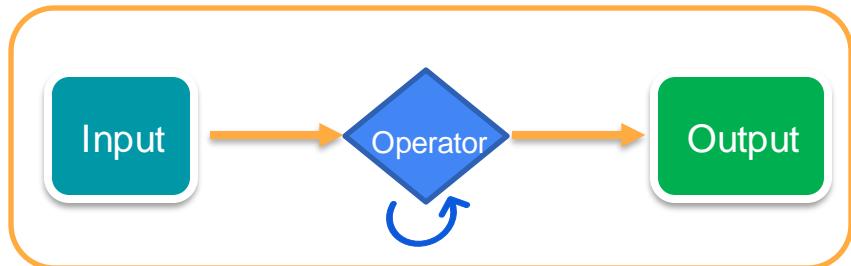
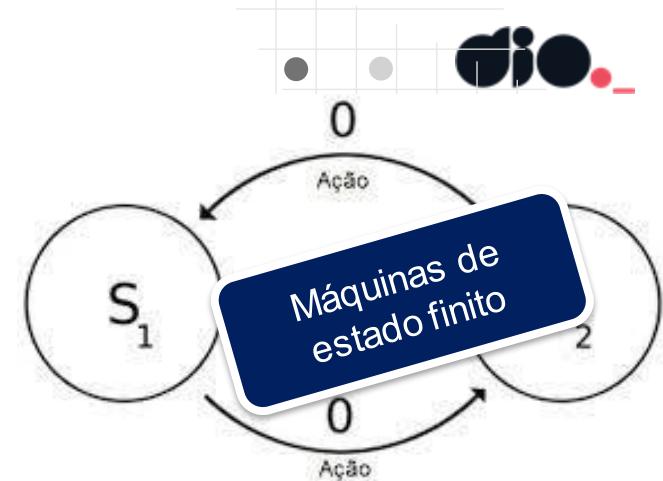
# Abstração



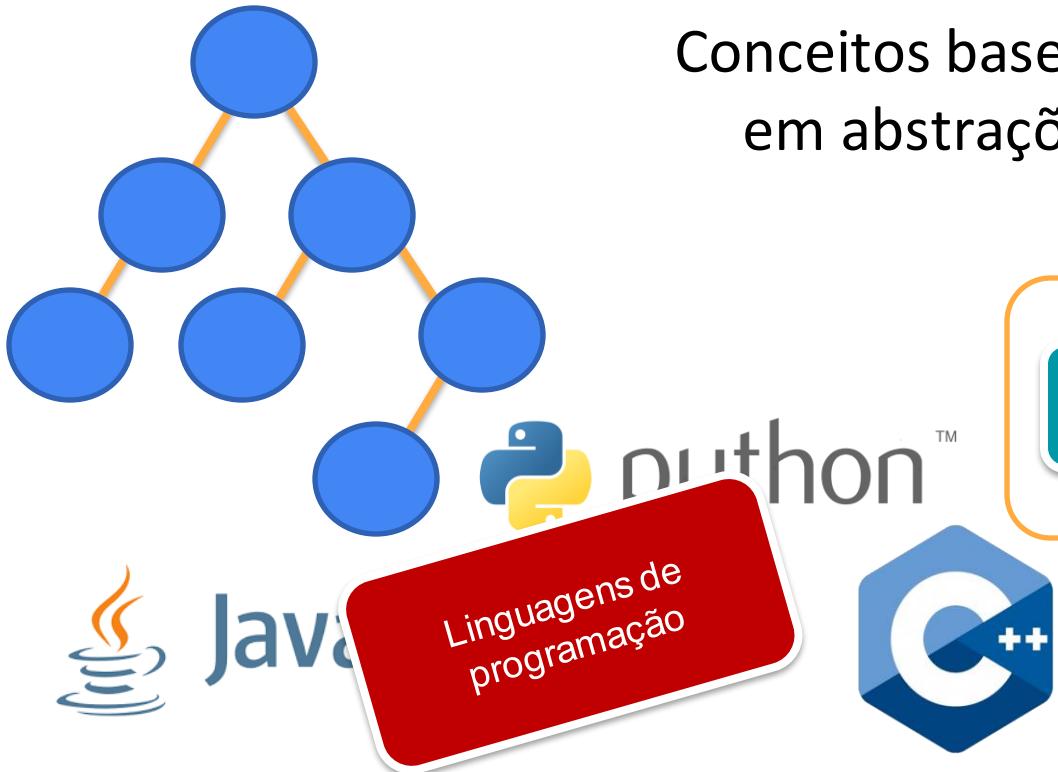
python™



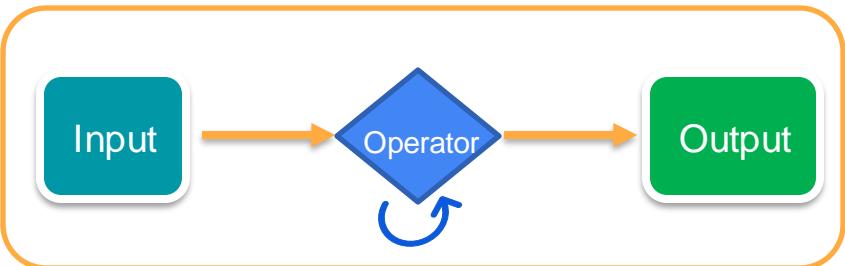
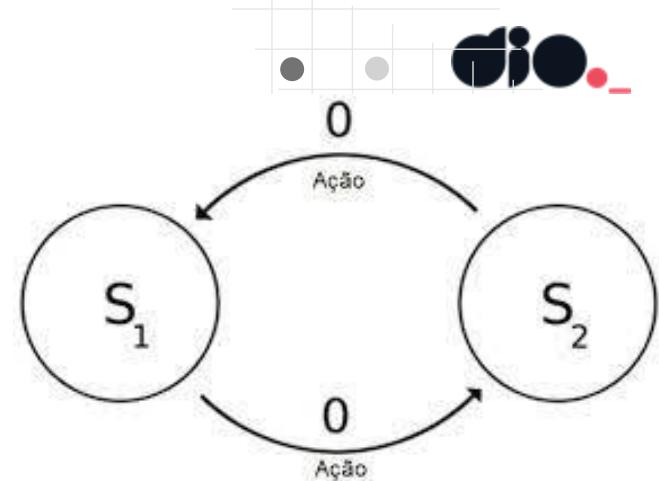
Conceitos baseados  
em abstrações



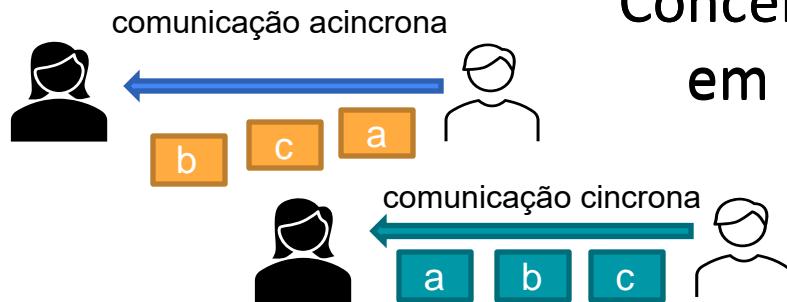
# Abstração



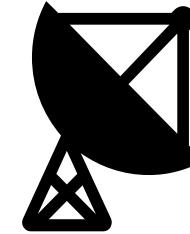
Conceitos baseados  
em abstrações



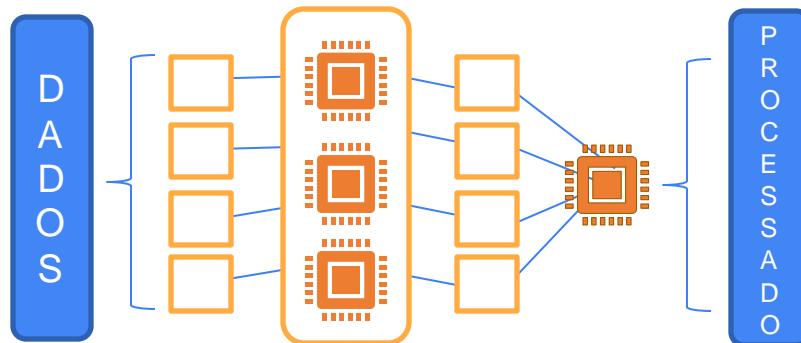
# Abstração



Conceitos baseados  
em abstrações



Paralelismo.

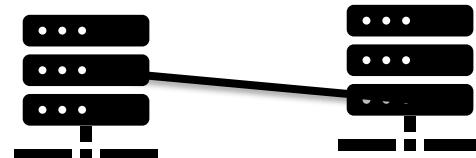
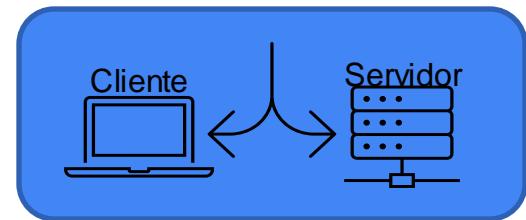
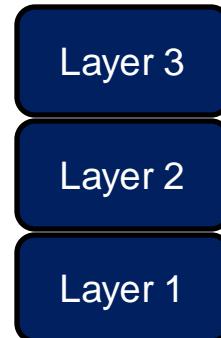


# Abstração

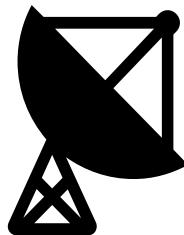
Redes protocolo TCP IP

Conceitos baseados  
em abstrações

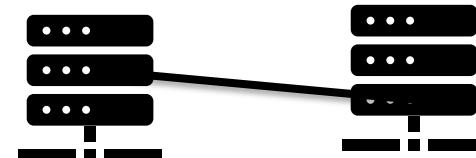
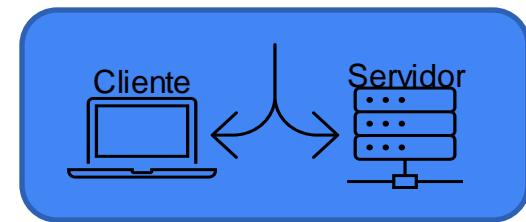
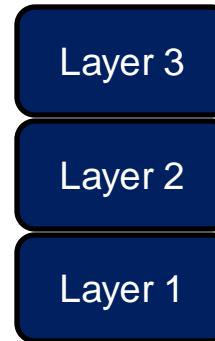
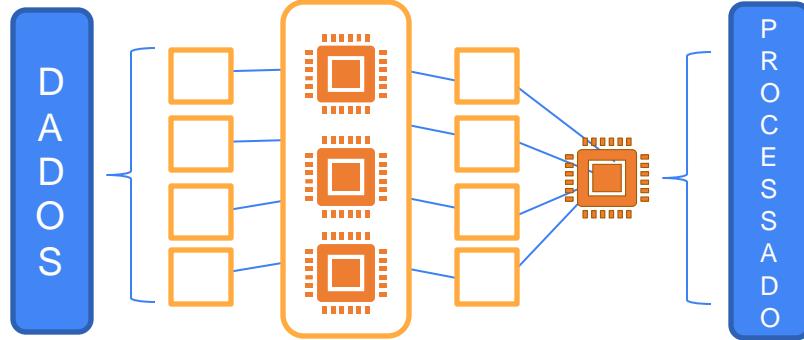
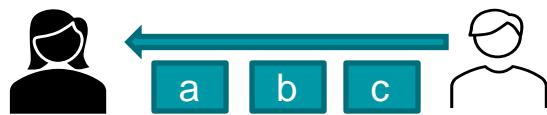
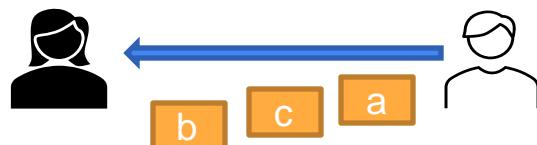
ARQUITETURAS



# Abstração



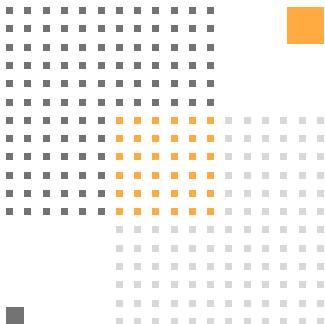
Conceitos baseados  
em abstrações



# Abstração

Conceitos baseados  
em abstrações

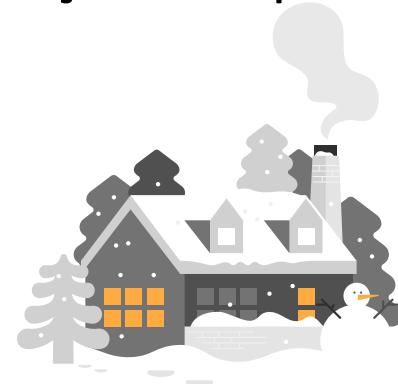
Exemplos



# Abstração



Por onde começar a limpar o terreno?

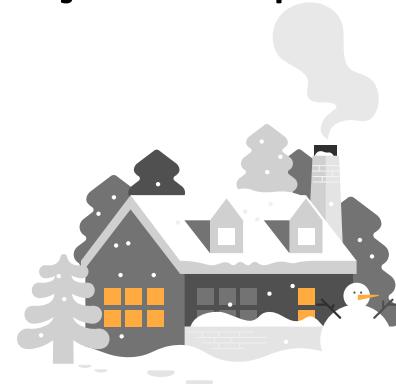


# Abstração

Classificar?



Por onde começar a limpar o terreno?



# Abstração

Classificar?



Por onde começar a limpar o terreno?

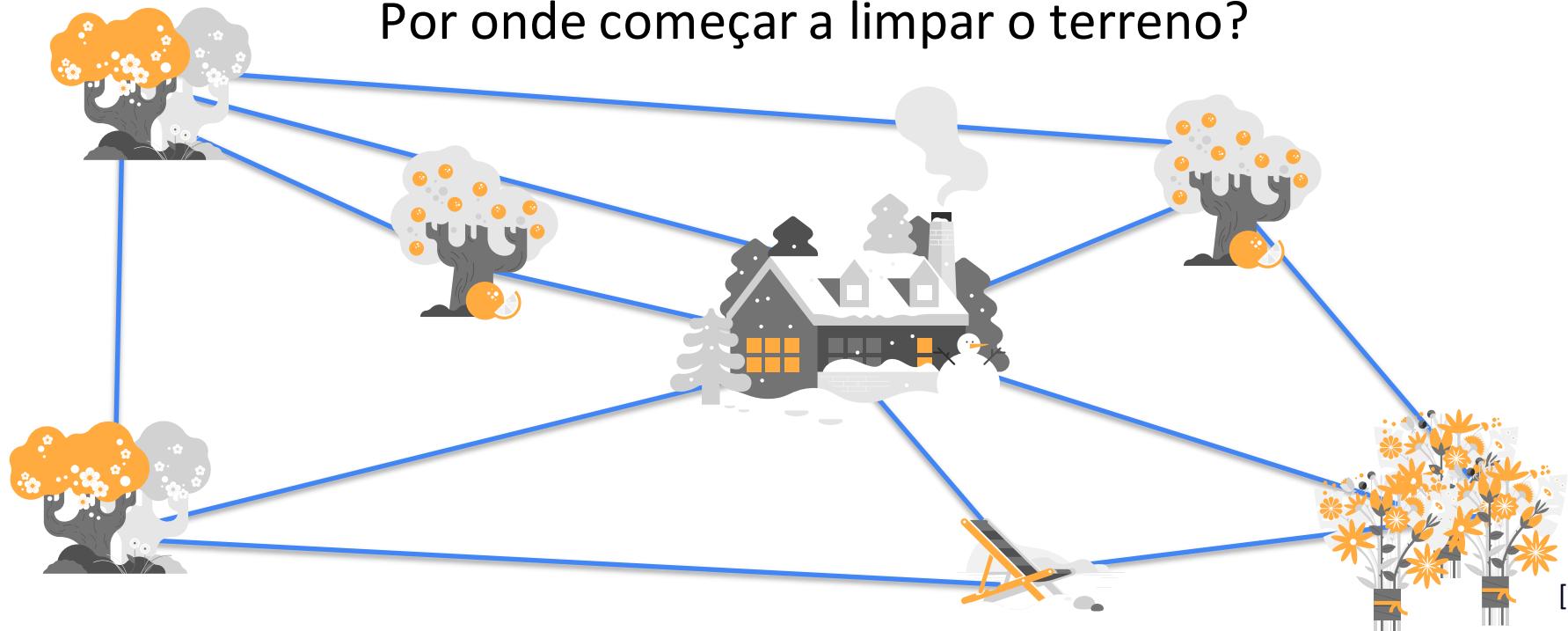


# Abstração

Distâncias



Por onde começar a limpar o terreno?

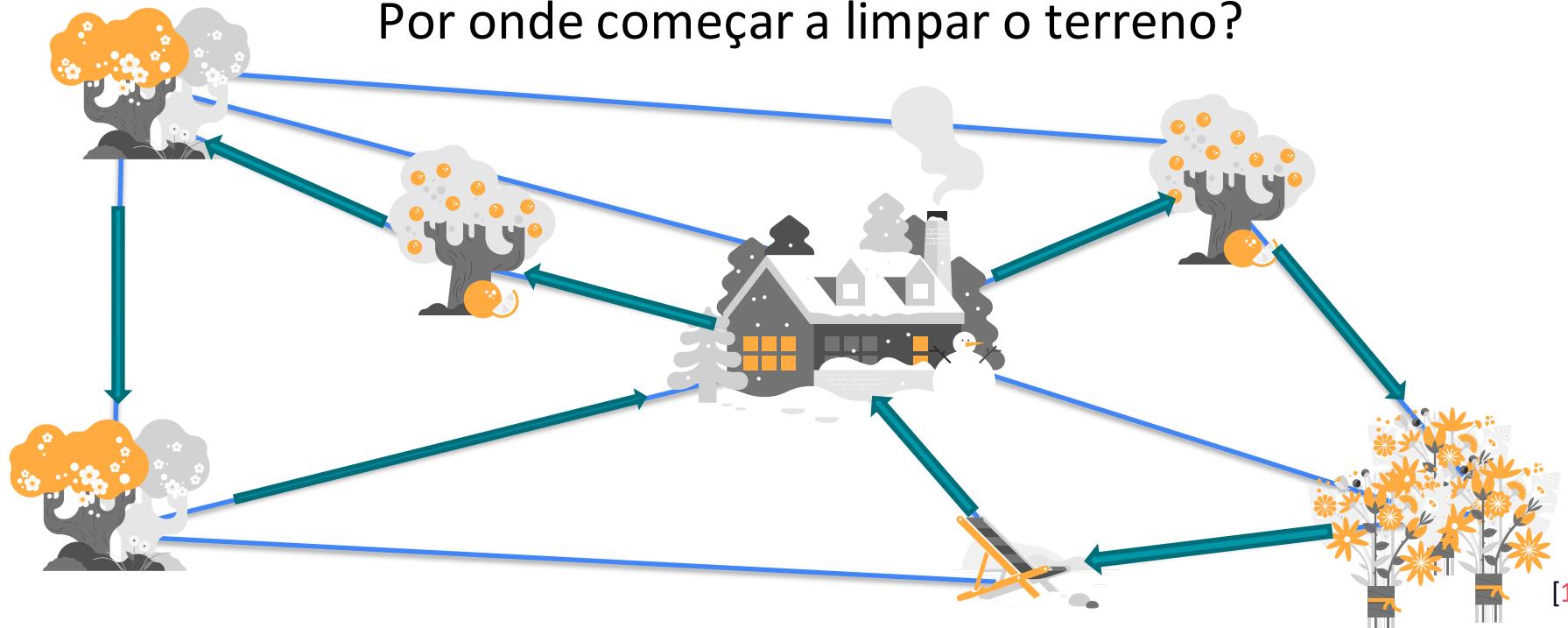


# Abstração

Distâncias



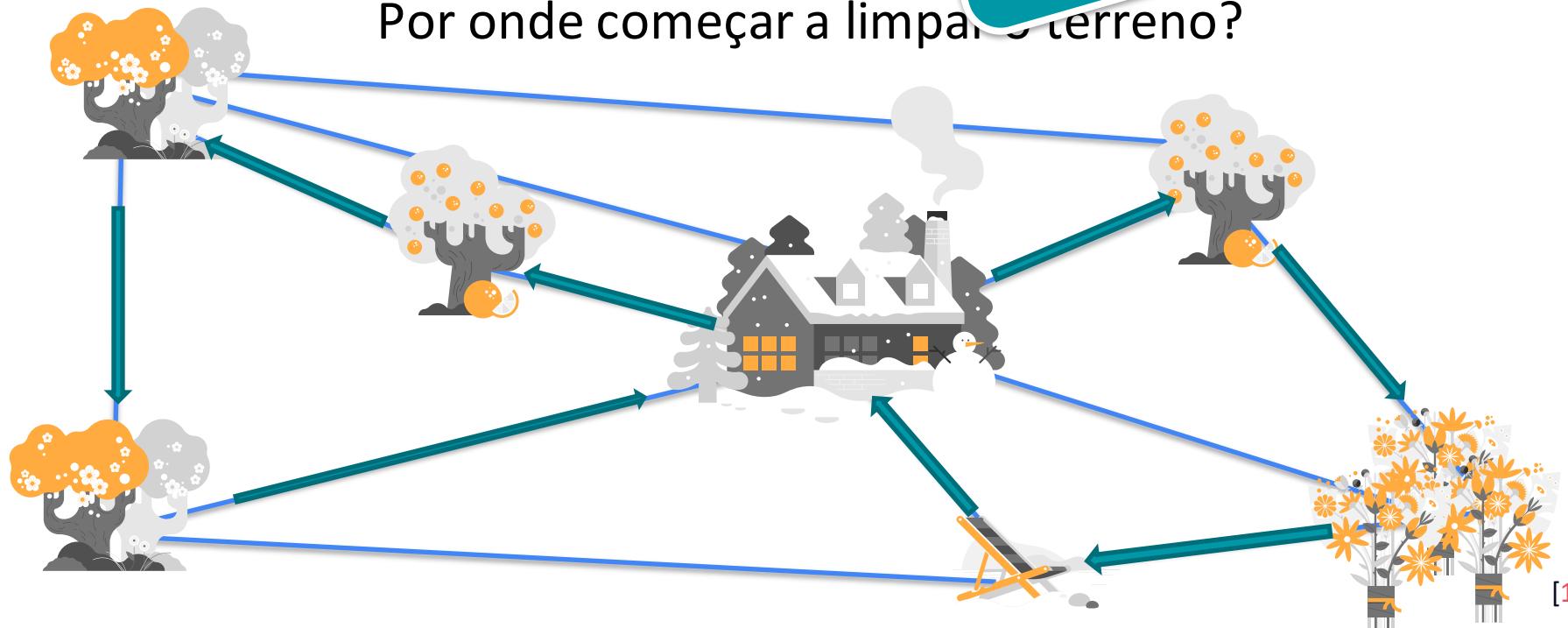
Por onde começar a limpar o terreno?



# Abstração

Estender para  
outros cenários

Por onde começar a limpar o terreno?

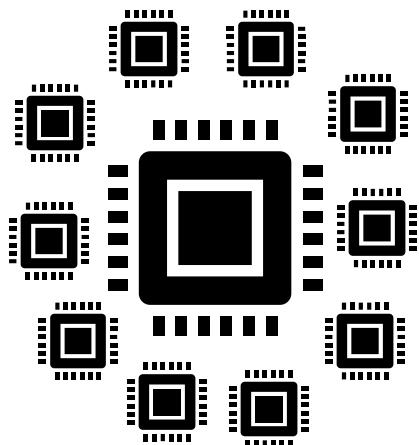


## Etapa 6

# Pilares: Algoritmos

// Primeiros passos para começar a  
programar/ Pensamento Computacional/

# Algoritmos



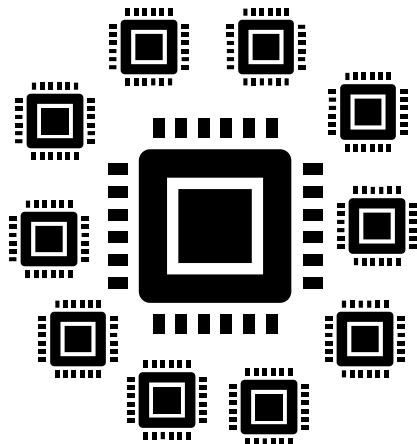
Tem:

- Energia
- Trabalhador
- Eficiência
- Rapidez

IOIO IOIO IOIO IOIO  
IOIO IOIO IOIO IOIO

# Algoritmos

tem que ser instruído.

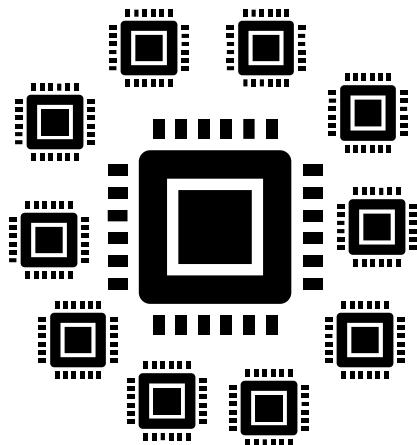


- Energia
- Trabalhador
- Eficiência
- Rapidez

NÃO OPERA  
SOZINHO

IOIO IOIO IOIO IOIO IOIO  
IOIO IOIO IOIO IOIO IOIO

# Algoritmos



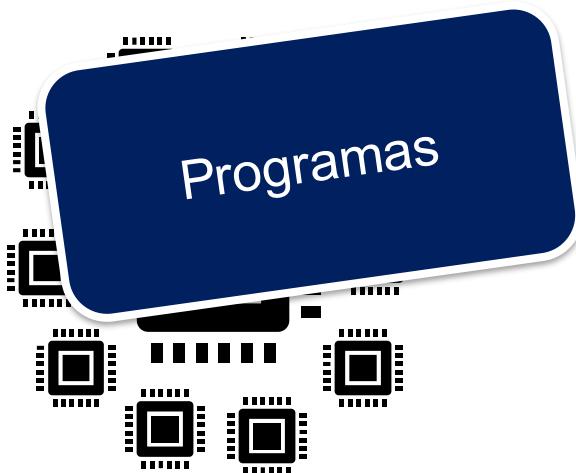
- Eficiência
- Trabalho em parceria
- Eficiencia
- Rapidez

PRECISA DE INSTRUÇÕES  
DETALHADAS

NÃO OPERA  
SOZINHO

IOIO IOIO IOIO IOIO IOIO  
IOIO IOIO IOIO IOIO IOIO

# Algoritmos

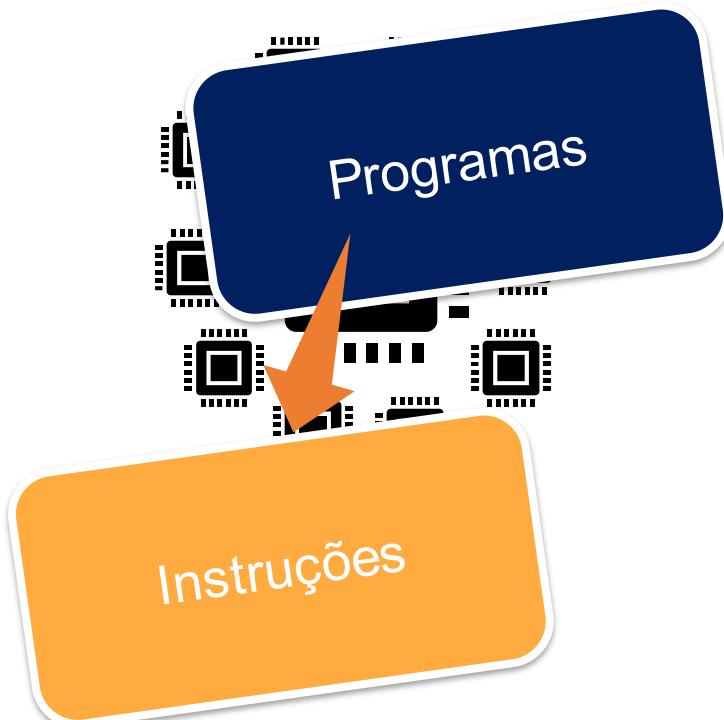


## PROCESSAMENTO DE DADOS

O computador recebe, manipula e armazena dados.

1010101010101010  
1010101010101010

# Algoritmos



## PROCESSAMENTO DE DADOS

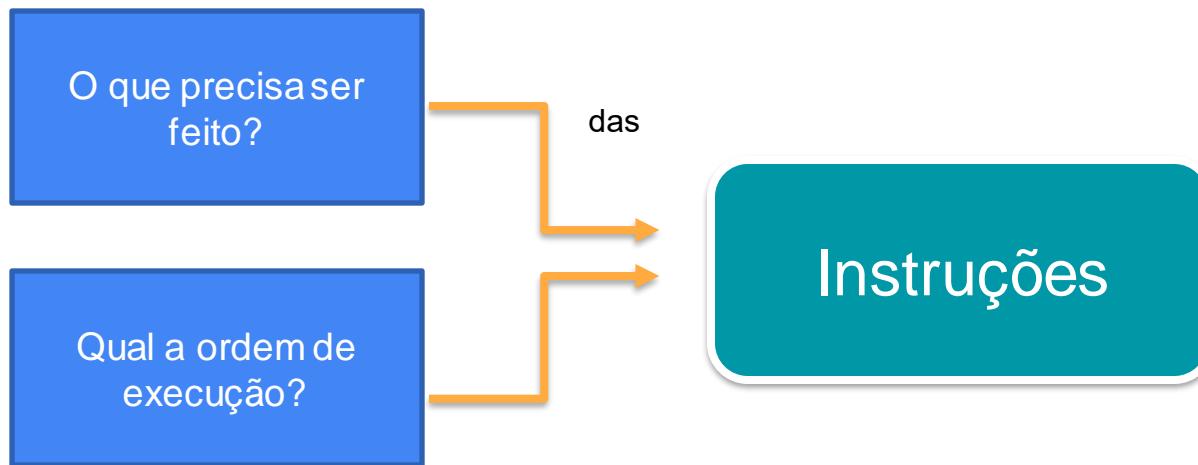
O computador recebe, manipula e armazena dados.

1010 1010 1010 1010  
1010 1010 1010 1010

Step by step porcesso passo a passo,

# Algoritmos

Processo de resolução de problemas "step by setp" utilizando instruções



# Algoritmos

O algoritmo tem que ser entendido por:

Processo de resolução

Entendimento por humano e máquina

"utilizando

O que é feito?

Qual a ordem de execução?

Instruções

# Algoritmos

## Desenvolvimento do Programa

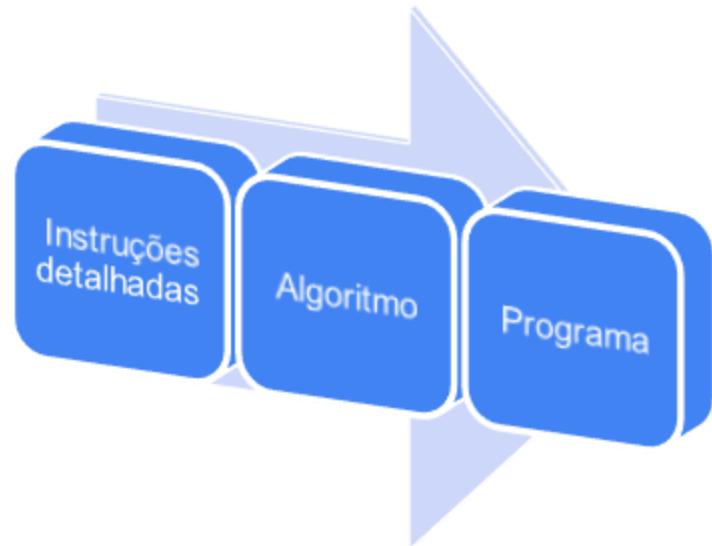
- Análise, entendimento dos casos.  
Estudo e definição dos dados  
de entrada e saída
- Algoritmo
- Codificação



# Algoritmos

## Desenvolvimento do Programa

- Análise
- Algoritmo
- Descreve o problema por meio de
- ferramentas narrativas, fluxograma,
- ou pseudocódigo
- Codificação



# Algoritmos

## Desenvolvimento do Programa

- Análise
- Algoritmo
- Codificação

O algoritmo é codificado de acordo com a linguagem de programação escolhida



# Algoritmos

É uma:

Sequência de passos com objetivo definido

Execução de tarefas específicas

Conjunto de operações que resultam  
em uma sucessão finita de ações

# Algoritmos

Preparar um sanduiche

Trocar uma lâmpada

Fazer uma receita de bolo

Trajeto ao trabalho

# Algoritmos

Instruções executadas passo a passo  
para concluir a tarefa

Preparar um sanduíche

Trajeto ao trabalho

Treino

Fazer uma receita de bolo

# Algoritmos

Como construir um algoritmo?

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

Pontos mais importantes

# Algoritmos

Como construir um algoritmo?

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

Dados fornecidos e  
Cenário

# Algoritmos

Como construir um algoritmo?

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

Cálculos e Restrições

# Algoritmos

Como construir um algoritmo?

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

Após processamento

# Algoritmos

Como construir um algoritmo?

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

Construção e refinamento do  
algoritmo

# Algoritmos

Construção de algoritmos

Narrativa

Fluxograma

Pseudocódigo

# Algoritmos

## Construção de algoritmos

Narrativa

Fluxograma

Pseudocódigo

Sem conceitos novos

Utilização da  
linguagem natural

Diversas interpretações possíveis

# Algoritmos

## Construção de algoritmos

Narrativa

Fluxograma

Pseudocódigo

Simples  
entendimento

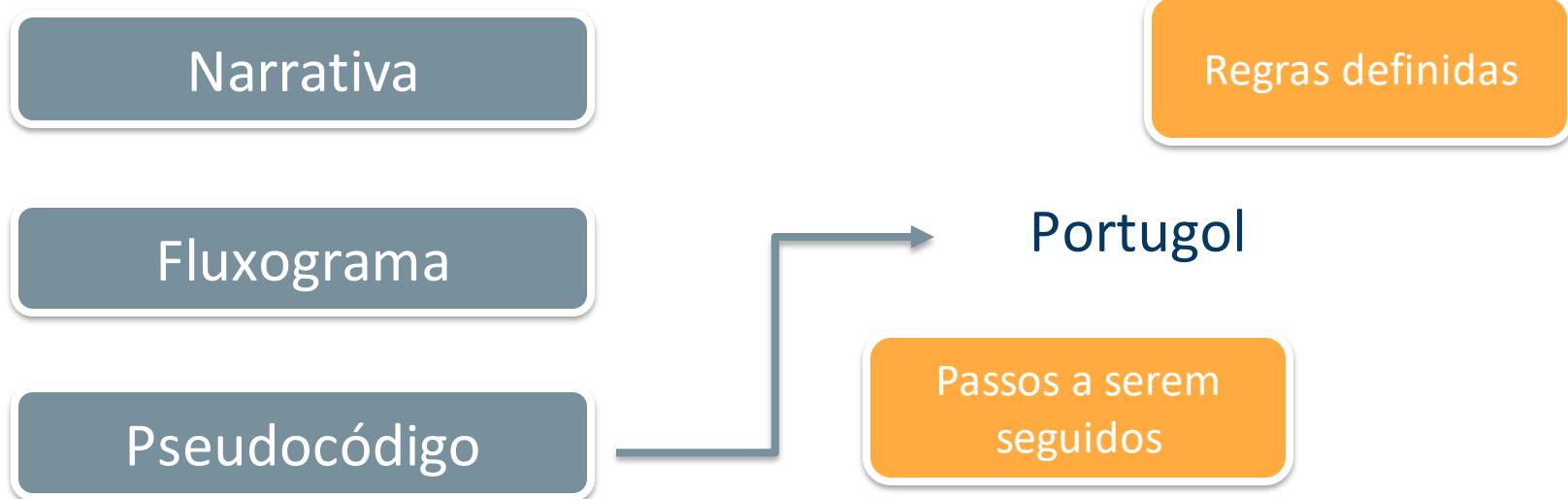
Utilização de símbolos  
pré-definidos

Conhecimento prévio da  
estrutura e símbolos



# Algoritmos

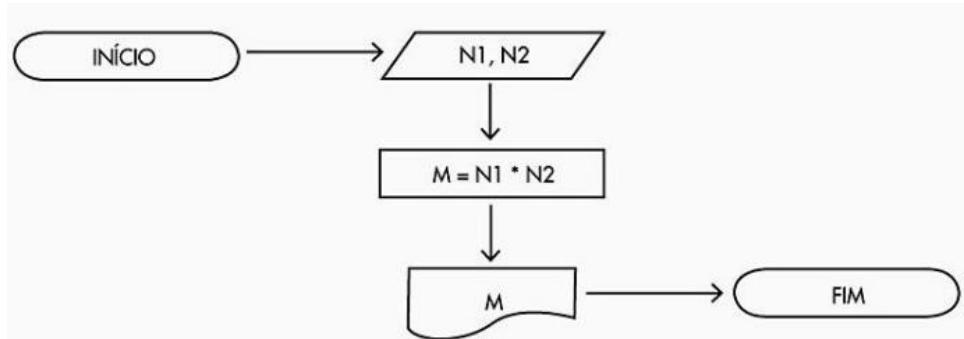
## Construção de algoritmos



# Algoritmos

Multiplicação de dois números:

- Passo 1 – recebe os valores
- Passo 2 – Multiplica
- Passo 3 – Imprime resultado



# Algoritmos

Média de alunos:

Passo 1 – recebe os valores

Passo 3 – Imprime resultado não realiza a operação

Passo 4 – Regra de aprovação

Passo 5 – Imprima o resultado



## Etapa 7

# Estudo de caso conceitual: perdido

// Primeiros passos para começar a  
programar/ Pensamento Computacional/

# Perdido



# Perdido

Como resolver o problema utilizando o pensamento computacional?



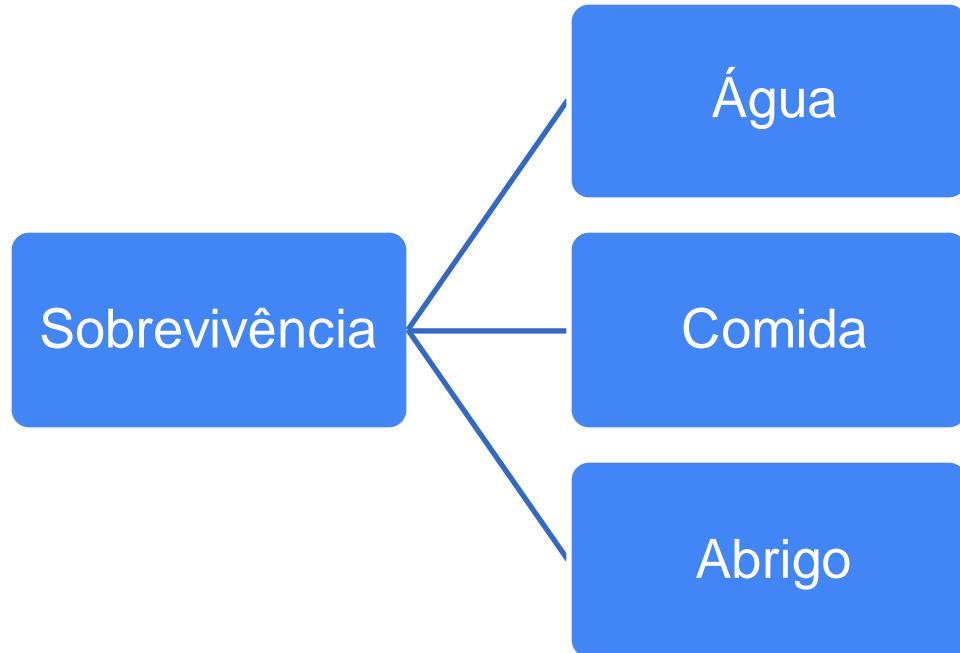
# Perdido

Como resolver o problema utilizando o pensamento computacional?

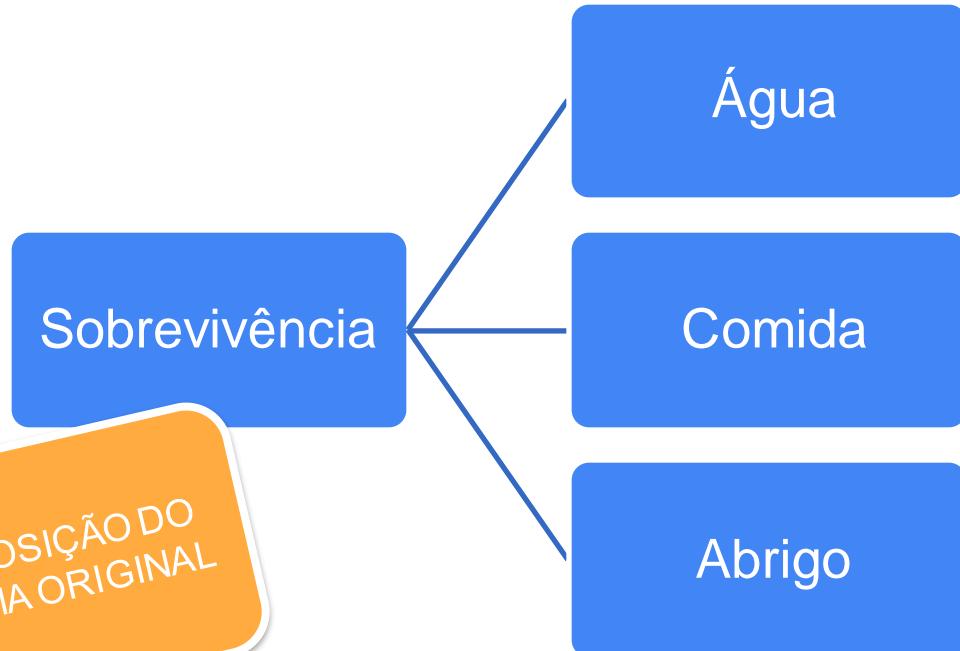
- Identificar mecanismos
- Recursos comuns
- Detalhes mais importantes



# Perdido



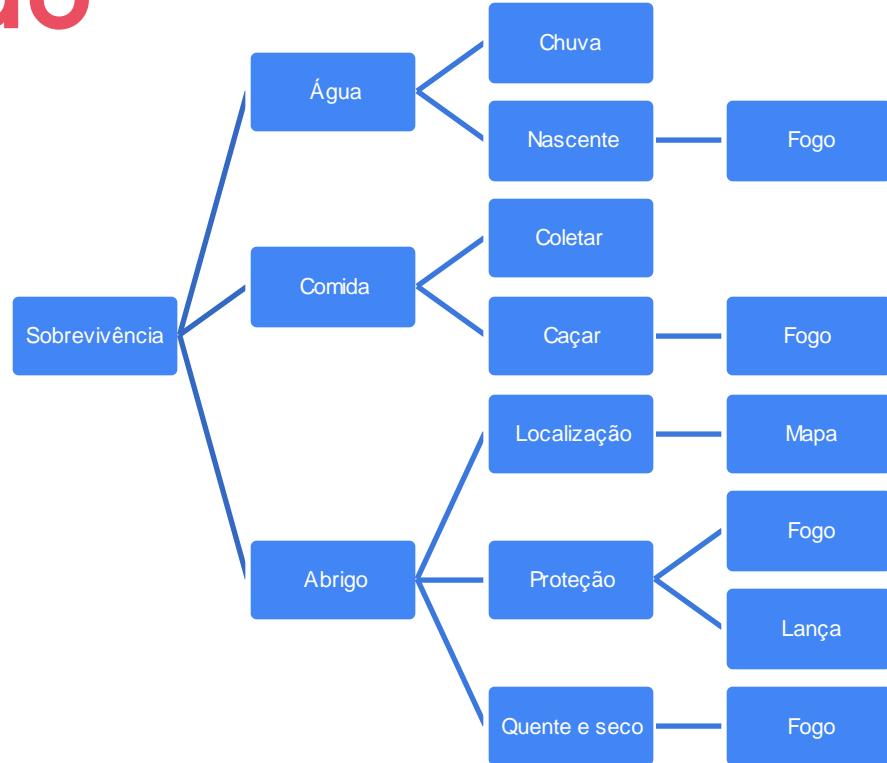
# Perdido



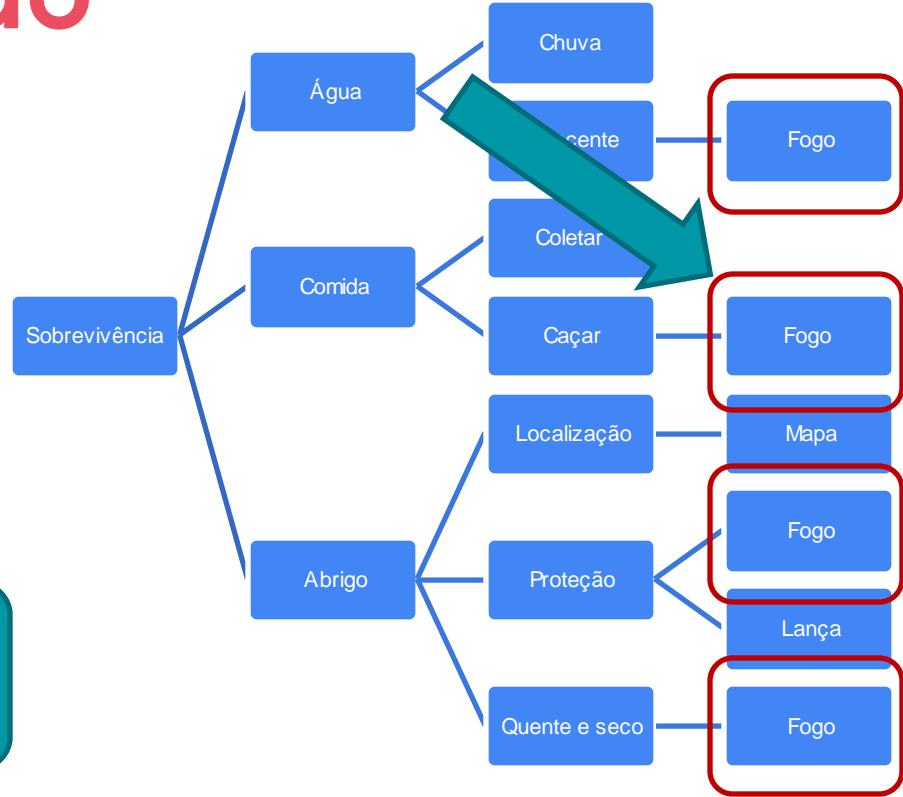
# Perdido



# Perdido



# Perdido



# Perdido

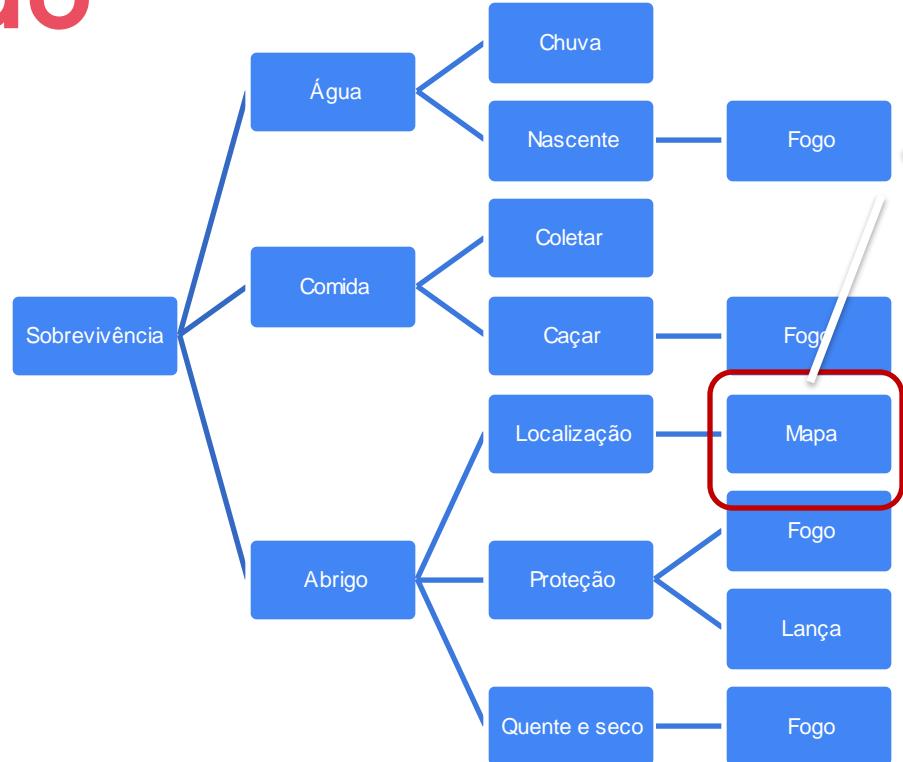


Onde construir o  
abrigos?

Danger zones



# Perdido



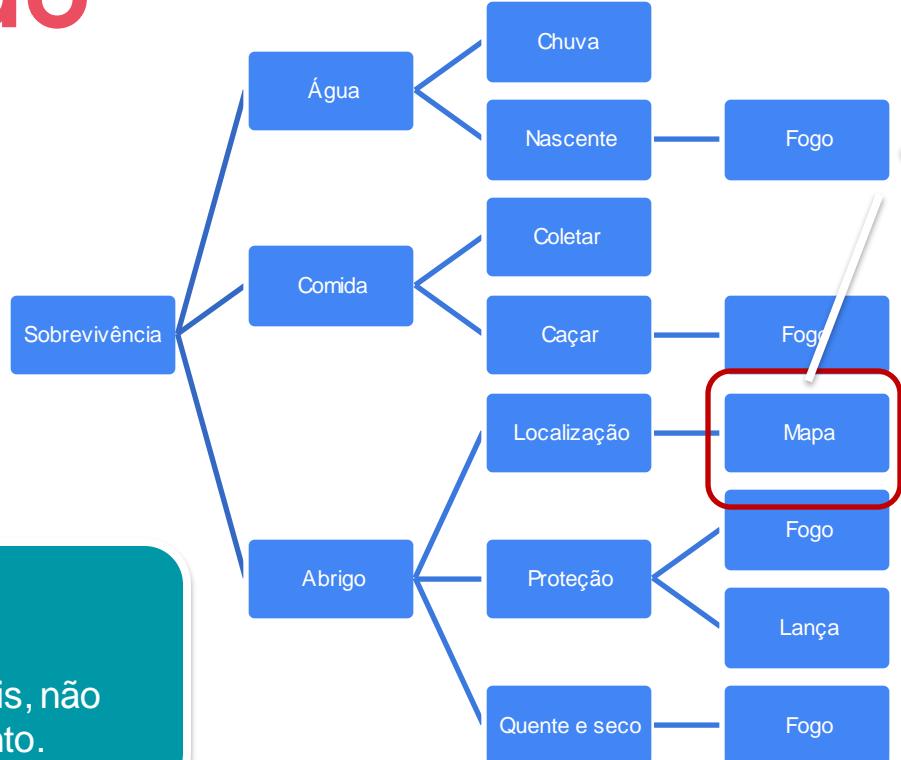
Criar por abstração



# Perdido

FOCO:

Nos aspectos principais, não precisa de detalhamento.



Criar por abstração



# Perdido

So far so good ...

- Decomposição
- Reconhecimento de padrões
- Abstração

Próximo passo ...



# Perdido

So far so good ...

Determinar instruções passo a passo para cozinhar

- Decomposição
- Reconhecimento de padrões
- Abstração

Próximo passo ...



# Perdido

Preparando a comida

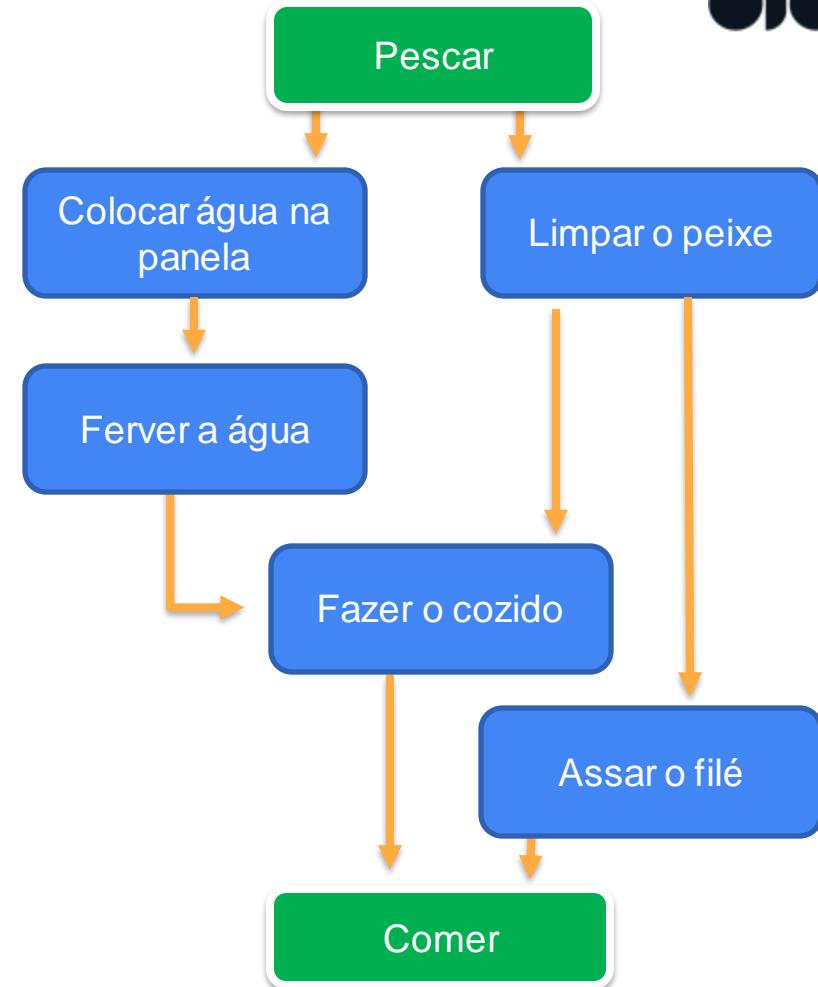
- Pegar o peixe
- Colocar água na panela
- Fever a água
- Limpar o peixe
- Fazer o cozido
- Assar o filé



# Perdido

## Preparando a comida

- Pegar o peixe
- Colocar água na panela
- Fever a água
- Limpar o peixe
- Fazer o cozido
- Assar o filé



# Perdido

Mesmo processo para

- Encontrar água
- Construir abrigo
- Maximizar chances de resgate
- ...



## Etapa 8

# Estudo de caso aplicado: soma de um intervalo

// Primeiros passos para começar a  
programar/ Pensamento Computacional/

# Soma de intervalo

Ex: soma de n° entre 1 e 200

1+2

1+3

1+4

1+5 ...

# Soma de intervalo

Ex: soma de n° entre 1 e 200

1+2

1+3

1+4

1+5 ...

Solução ineficiente!

# Soma de intervalo

Ex: soma de n° entre 1 e 200

$$200 + 1$$

$$199 + 2$$

$$198 + 3$$

$$197 + 4 \dots$$

Outra forma de  
expressar a solução

# Soma de intervalo

Ex: soma de nº entre 1 e 200

$200 + 1$

$199 + 2$

$198 + 3$

$197 + 4 \dots$

$200 + 1 = 201$

$199 + 2 = 201$

$198 + 3 = 201$

$197 + 4 = 201 \dots$



Padrão

Decomposição

# Soma de intervalo

Ex: soma de n° entre 1 e 201

$$200 + 1$$

$$199 + 2$$

$$198 + 3$$

$$197 + 4 \dots$$

Como expressar de forma  
generalista?

$$1 = 201$$

$$199 + 2 = 201$$

$$198 + 3 = 201$$

$$197 + 4 = 201 \dots$$

Decomposição

Padrão

# Soma de intervalo

Ex: soma de n° entre 1 e 200

$$200 + 1 = 201$$

$$199 + 2 = 201$$

$$198 + 3 = 201$$

$$197 + 4 = 201$$

...



Valor se repete



Quantas vezes?

$$200/2 = 100$$

Resultado

$$201 \times 100 = 20.100$$

# Soma de intervalo

Ex: soma de n° entre 1 e 200

Expressar em variáveis

$$200 + 1 = 201$$



Valor se repete

$$199 + 2 = 201$$



$$198 + 3 = 201$$

Quantas vezes?

$$200/2 = 100$$

$$197 + 4 = 201$$

...

Resultado

$$201 \times 100 = 20.100$$

# Soma de intervalo

Ex: soma de n° entre x e y

$[x, y] \rightarrow$  intervalo



1 e 200

Expressar em variáveis

# Soma de intervalo

Ex: soma de n° entre x e y

$[x, y] \rightarrow$  intervalo

1 e 200

Expressar em variáveis

$y+x =$  resultado parcial  
 $(y-1) + (x+1) =$  resultado\_parcial

...

$200 + 1 = 201$   
 $199 + 2 = 201$

# Soma de intervalo

$$100 \times 201 = 20.100$$

Ex: soma de n° entre x e y

$[x, y] \rightarrow$  intervalo

Expressar em variáveis

total x resultado parcial = resultado

$$y+x = \text{resultado parcial}$$

$$(y-1) + (x+1) = \text{resultado\_parcial}$$

...

$$y/2 = \text{total}$$

$$200/2 = 100$$

# Soma de intervalo

Passo 1 – Recebe os valores (x e y)

Passo 2 – Resolva:

$$y/2 = \text{total}$$

Passo 3 – Resolva:

$$y+x = \text{resultado\_parcial}$$

Passo 4 – Ache o total

$$\text{Final} = \text{total} \times \text{resultado\_parcial}$$

Passo 5 – Imprima o resultado

Algoritmo

## Etapa 9

# Estudo de caso aplicado: adivinhe o número

// Primeiros passos para começar a  
programar/ Pensamento Computacional/

# 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.

# Adivinhe o número

Ex: adivinhe o número



O número é 1?

O número é 2?

O número é 3?

...

Possível solução

# Adivinhe o número

Ex: adivinhe o número



O número é 1?

O número é 2?

O número é 3?

Ineficiente

...

Possível solução

Desperdício de tempo

# Adivinhe o número

Ex: adivinhe o número



Possível solução

P: O número é maior que 50?

R: Não.

P: O número é menor que 20?

R: Sim.

Nº de tentativas  
muito menor

...

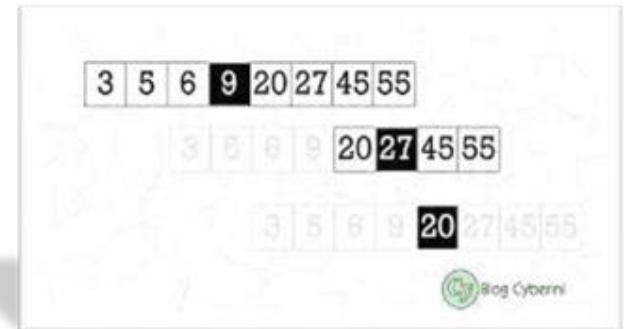
até encontrar o número

# Adivinhe o número

Ex: Busca binária

Método de busca

Mais eficiente que busca por varredura



# Adivinhe o número

Método de busca

Ex: Busca binária

Mais eficiente que busca por varredura

Passo 1 – Ordenar o vetor

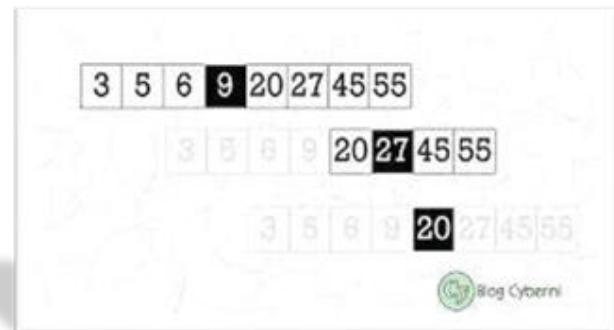
Passo 2 – Módulo de L/2

Passo 3 – Acessar estrutura

Passo 4 – Comparar valores

Passo 5 – Repita até encontrar o número

Passo 6 – Imprima "Busca bem sucedida"



# Habilidade

Como aprimorar essa habilidade?

Permitindo que seus alunos expliquem suas decisões e seu processo de desenvolvimento ...

(Brennan & Renesck)

Para um problema proposto

# Aula 2: Introdução à Lógica de programação

// Primeiros passos para começar a programar

# Objetivo Geral

Esta aula foca em apresentar o conceito de lógica aplicada à programação, como um processo de pensamento atrelado ao conceito de algoritmos e resolução de problemas.

# Percurso

**Etapa 1**

O que é Lógica?

**Etapa 2**

Técnicas de lógica da programação

**Etapa 3**

Breve história da computação

## Etapa 1

# O que é Lógica?

// Primeiros passos para começar a programar/  
Introdução à Lógica de Programação

# Lógica

02 O problema é uma

Questão

01 A lógica é para  
solucionar um

Problema

04 pode surgir

Proposta duvidosa

Na resolução do problema

O problema pode ter

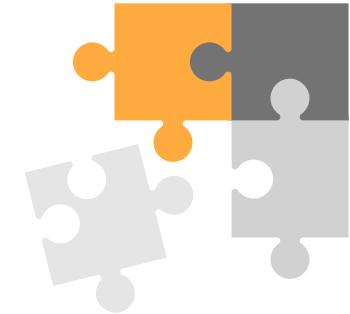
Numerosas soluções

03 O problema também é um

Objeto de discussão

# Lógica

## Definição formal



"Problema é uma questão que foge a uma determinada regra, ou melhor é um desvio de percurso, o qual impede de atingir um objetivo com eficiencia e eficácia."



# Lógica

Lógica



# Lógica

## Definição formal de lógica



"Parte da filosofia que trata das formas do pensamento em geral (dedução, indução, hipótese, inferência etc.) e das operações intelectuais que visam à determinação do que é verdadeiro ou não."



# Lógica

04

Forma como desencadeiam  
acontecimentos

02 A lógica é uma

Forma de raciocínio

Lógica

01 A lógica é uma

Organização coesa

03 Onde teremos uma

Ordenação que segue convenções



# Lógica

Significado computacional

Lógica



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

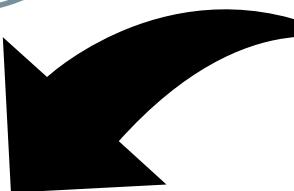
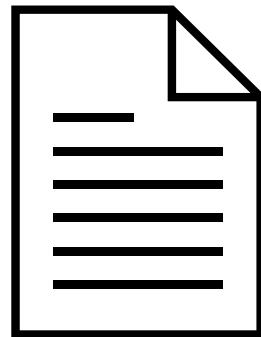
# Lógica

Percebe que essa  
definição se aplica no  
seu cotidiano?



# Lógica

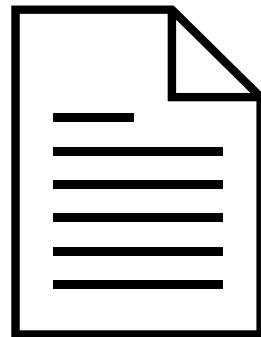
Percebe que essa definição se aplica no seu cotidiano?



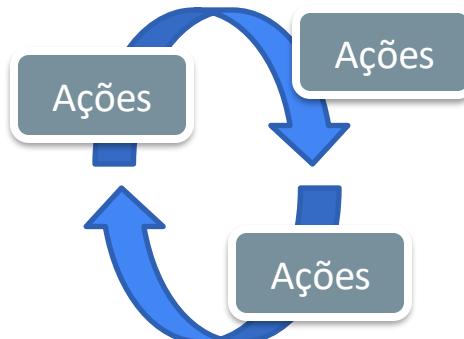
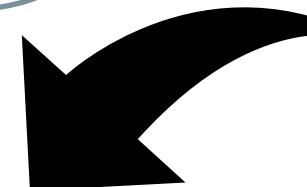
Sequência de instruções

# Lógica

Percebe que essa definição se aplica no seu cotidiano?

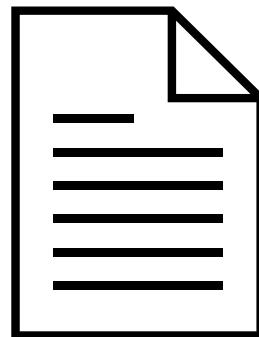


Sequência de instruções



# Lógica

Percebe que essa definição se aplica no seu cotidiano?



A lógica gera uma

Sequência de instruções

Instruções que são baseadas em

Ações

Ações

Ações

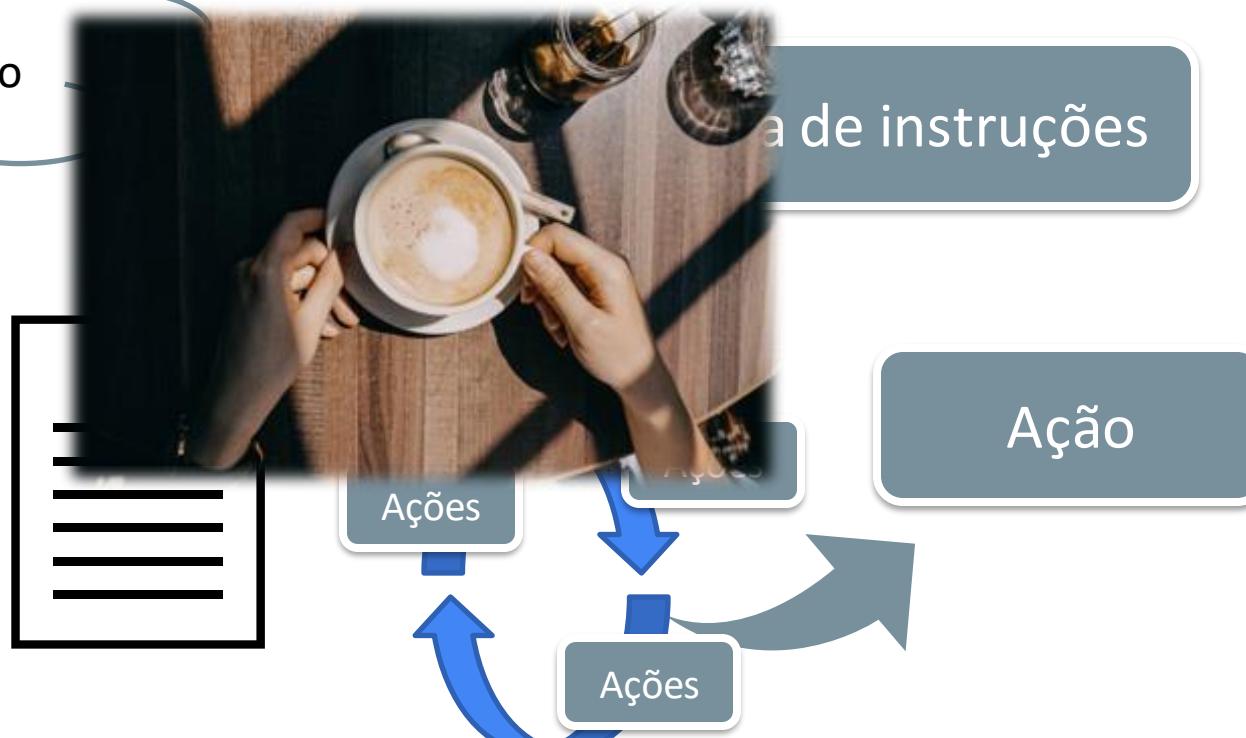
Ação



Estas ações levam a uma ação maior e principal que é o nosso objetivo.

# Lógica

Percebe que essa definição se aplica no seu cotidiano?

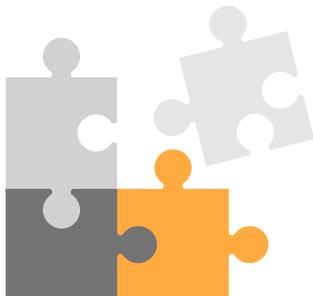


# Lógica

Falta de costume



Seres humanos podem prever  
comportamentos, computadores não.



# Lógica

Falta de costume



Seres humanos podem prever comportamentos, computadores não.



Instruções detalhadas



# Lógica

Por que entender a lógica  
em programação?

"Everybody in this country should learn  
how to program a computer because it  
teaches you how to think."

Todos deveriam aprender a programação  
de computadores, pois isso ensina Steve Jobs  
como pensar.



## Etapa 2

# Técnicas de Lógica de Programação

// Primeiros passos para começar a programar/  
Introdução à Lógica de Programação

# Técnicas



Planta baixa

# Técnicas



Planta baixa



Projeto

# Técnicas



Planta baixa



Projeto



Planejamento

# Técnicas

tem: o



Planta baixa

Todos

Projeto

Seguem a lógica e determinam as instruções



Planejamento

sequênciais

# Técnica Linear

- Modelo tradicional
- Não tem vínculo
  - Estrutura hierárquica
  - Programação de computadores

Modelos de desenvolvimento e resolução

# Técnica Linear

- Modelo tradicional
- Não tem vínculo
  - Estrutura hierárquica
  - Programação de computadores

é uma técnica usada na



fonte:  
editoraconstruir.com.br

Devido as características  
linear.

Técnica linear é um

Modelos de desenvolvimento e resolução

# Técnica Linear

Tem uma

Execução sequenciada



fonte:  
[editoraconstruir.com.br](http://editoraconstruir.com.br)

Ordenação de elementos por uma única  
propriedade.

tem

Recursos limitados

tem uma

Única dimensão

Modelos de desenvolvimento e resolução

# Técnica Linear



# Técnica Linear



Acordar



# Técnica Linear



Exemplo

# Técnica Linear



Istruções passo a passo.

# Técnica Estruturada

Estrutura

Organização, disposição e ordem dos elementos essenciais que compõem um corpo (concreto ou abstrato)

Modelos de desenvolvimento e resolução

# Técnica Estruturada

Objetivo

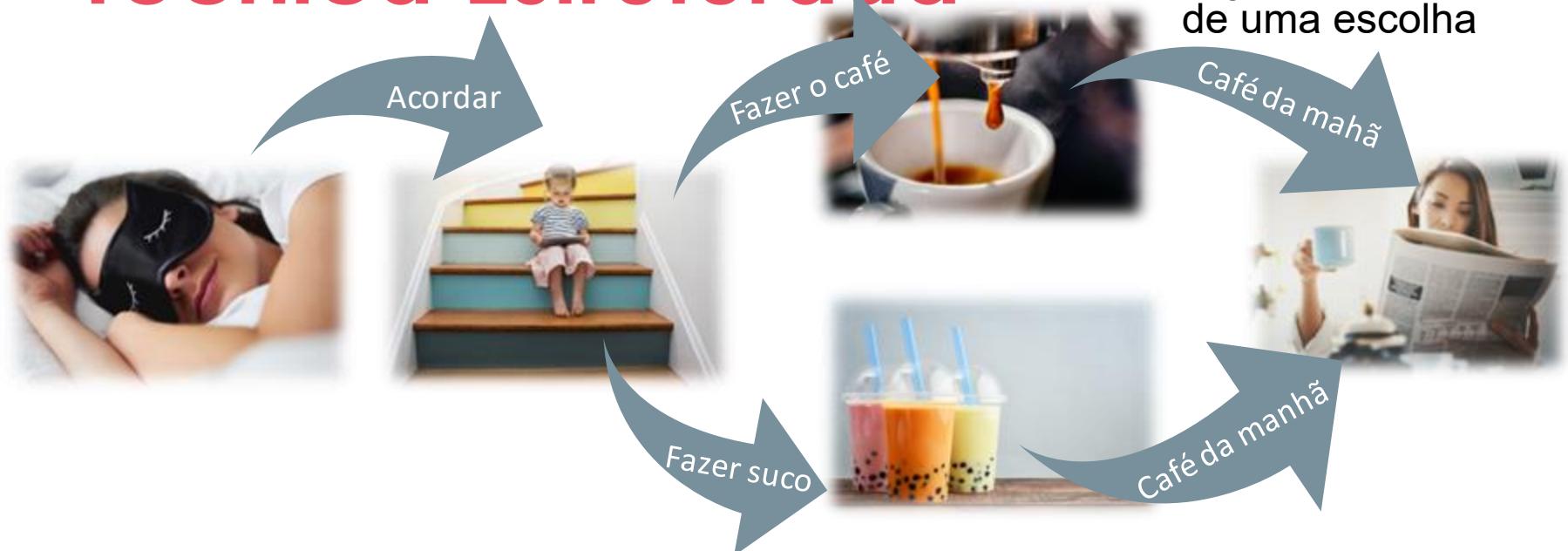
Processamento de dados



Modelos de desenvolvimento e resolução

Facilitar

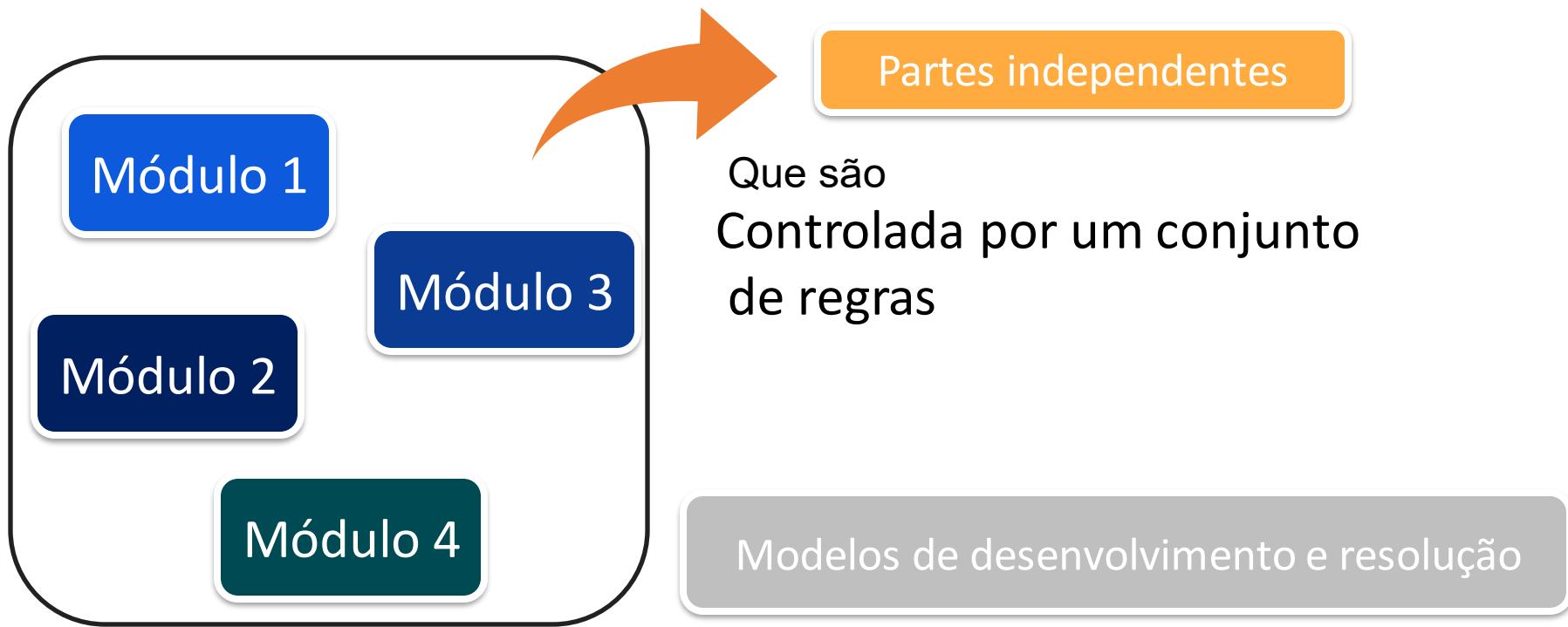
Na **Técnica Estruturada** tem-se condições pode haver mais de uma escolha



Modelos de desenvolvimento e resolução

Obs. na técnica linear não tem condições e nem escolhas

# Técnica Modular



Dentro da

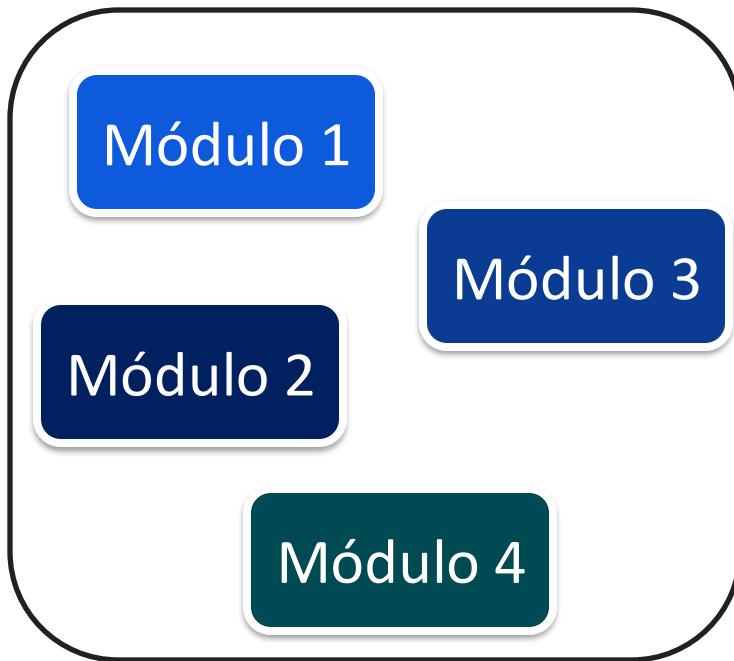
# Técnica Modular

tem:



Modelo padrão

# Técnica Modular



Metas

O que se tem com uma  
técnica modular:

- Simplicação
- Decompor o problema
- Verificação do módulo

Modelos de desenvolvimento e resolução

# Técnica Modular

Módulo preparar  
acordar



Módulo preparar bebida



Módulo tomar  
café da manhã



# Aula 3: Fundamentos de algoritmos

// Primeiros passos para começar a programar

# Objetivo

Nesta aula serão apresentados os conceitos básicos para o correto entendimento de algoritmos.

Esses conceitos como, variáveis, tipos de dados, instruções, condições, entre outros temas relacionados.

# Percurso

**Etapa 1** Tipologia e variáveis

**Etapa 2** Instruções primitivas

**Etapa 3** Estruturas condicionais e operadores

Obs. usada para guardar valores, ela recebe um valor, que tem que ser igualado ao valor que queremos armazenar.

[231]

# Percurso

Etapa 4

Estruturas de repetição

Etapa 5

Vetores e matrizes

Etapa 6

O que são funções?

# Percurso

**Etapa 7** Instruções de entrada/saída

## Etapa 1

# Tipologias e variáveis

// Primeiros passos para começar a programar/  
Fundamentos de algoritmos

# Variáveis

Qual a função do computador?



# Variáveis

Qual a função do computador?

É a de processar

Informação

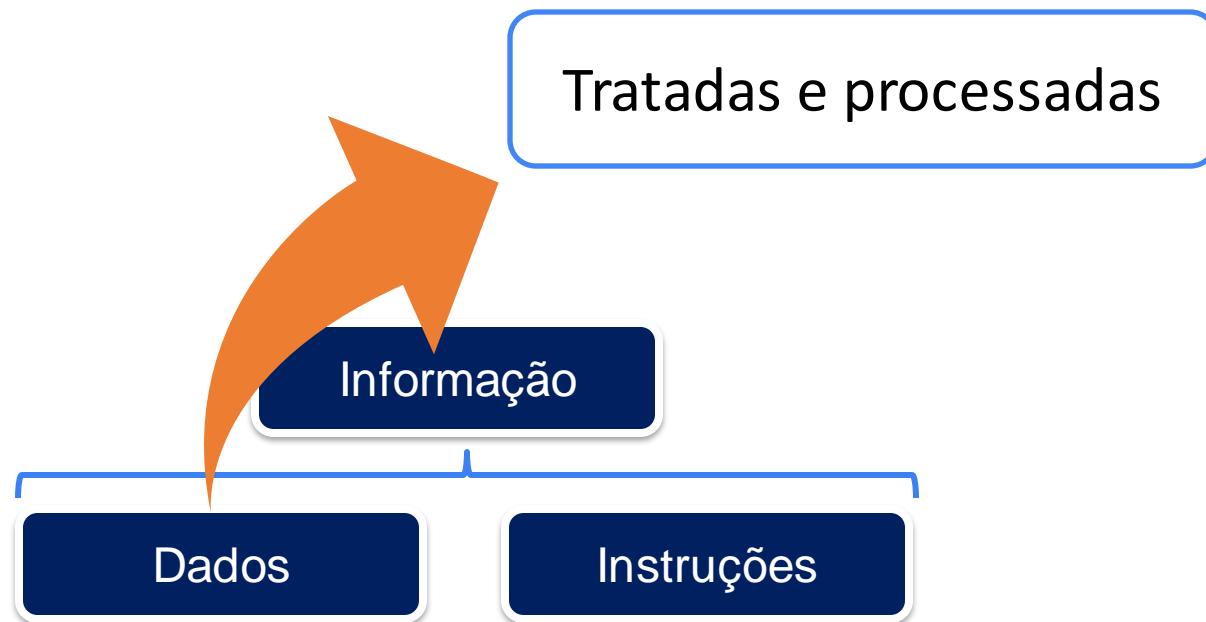
que são compostas por:

Dados

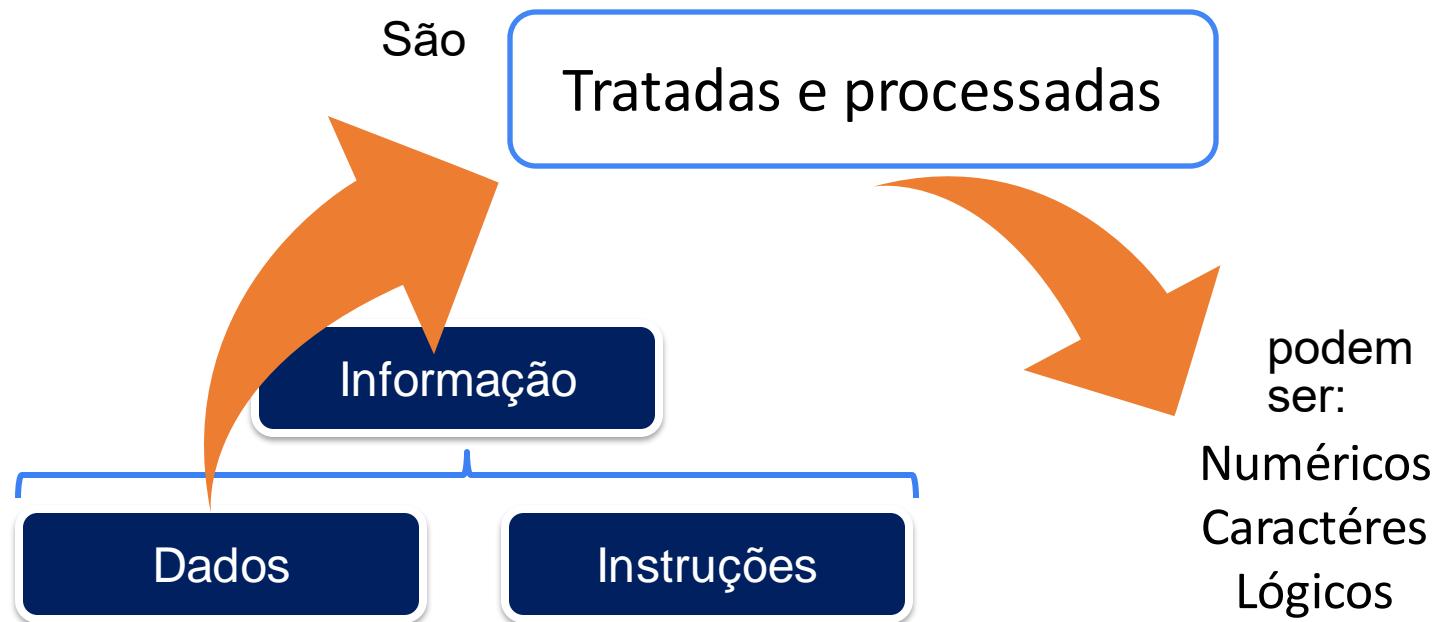
Instruções



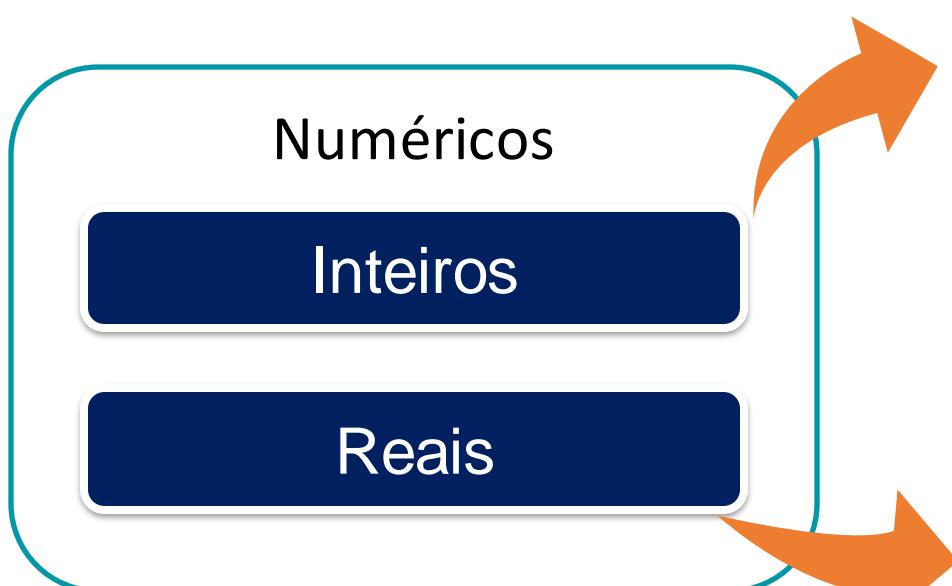
# Tipos de dados



# Tipos de dados



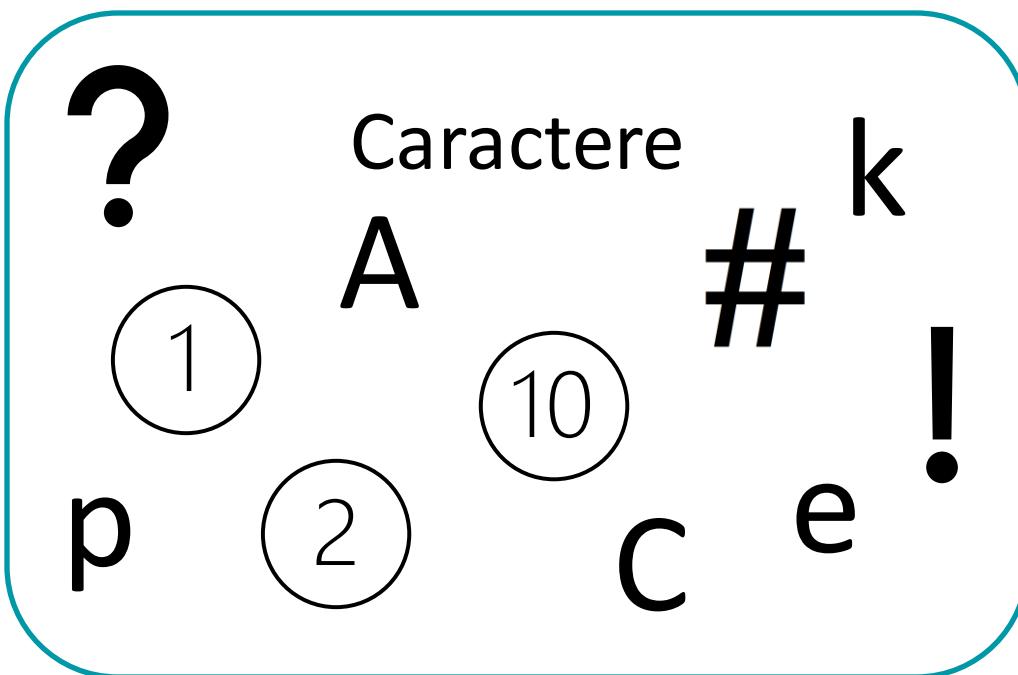
# Tipos - Numérico



0, 1, 5, 50, 60 , 800, ...  
-58, -50, -49, 32, -10, -5

5.95, 9.54, -8.8, -0.555 ...  
0, 1, 5, 50, 60 , 800, ...  
-58, -50, -49, 32, -10, -5

# Tipo - Caracteres



"Programação"  
"PROGRAMAÇÃO"  
"KU\*&NH53g"

"Fone:(25)99865-5741"  
"Rua alameda, n°45"

# Tipo - Lógico

Lógico

Verdadeiro - 1

Falso - 0

Booleano

# Tipo - Lógico

Lógico

Verdadeiro - 1

Falso - 0

Booleano

.Verdadeiro .V ou .S  
.Falso .F ou .N

Português estruturado

# Variável

O que é uma variável?



Possui variações

Incerto

Mutável

Inconstante

Instável

Obs: o valor tem que ser igual ao tipo de dado declarado .

Obs. usada para guardar valores, ela recebe um valor.

# Variável

O que é uma variável?

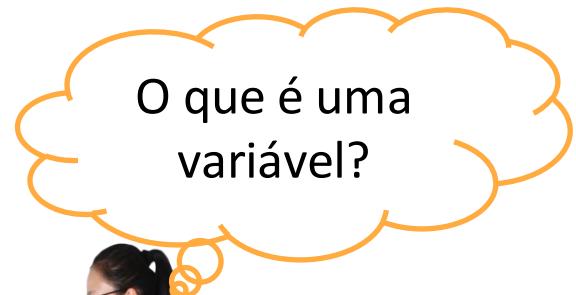


Variável - pode assumir qualquer um dos valores de um determinado conjunto de valores

$$a - b = d$$

$$a + b = c$$

# Variável



Variável - pode assumir qualquer um dos valores de um determinado conjunto de valores



$$a - b = d$$

$$a + b = c$$

# Variável

A Variável seria como uma gaveta ou uma caixa dentro de um armário.



# Variável

Que vai

Identificar

Variável

o dado é o tipo de  
dado que vai ser  
usado



# Variável

Nome da variável

## Regras

- Atribuição de um ou mais caracteres
- Primeira letra - não número, não pode ser número
- Sem espaços em branco
- Vedado Não é permitido a:
  - Utilização de palavras reservadas
- Caracteres e números

Exemplos de

# Variável

Regras

x2

Nome da variável

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

Nome\_usuario

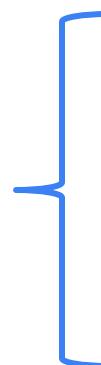
telefone

z4

user12

# Variável

Papéis de uma variável



A variável modifica o algoritmo.



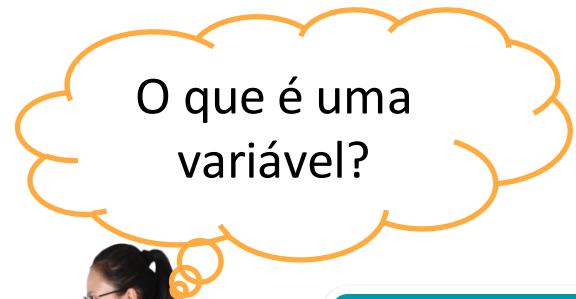
Modificação de estado

Controle

Viagiada, controle

A variável controla uma estrutura dentro do algoritmo ou uma equação.

# Variável



O que é uma  
variável?

Inalterável

Definição

Tudo aquilo que é fixo ou estável.

Invariável

O que não muda

# Constante Variável



Inalterável

Definição

Tudo aquilo que é fixo ou estável.

Invariável

O que não muda

# Costante

## ~~Variável~~

Constante?

Exemplos

O que não muda



Inalterável

Definição

$\pi$

pi = 3,14

é aquilo que é fixo ou estável.

$\Phi$

phi = 1,618

resulta = recebido \* 0.10

Invariável

# Objetivo

Os dados são

Tratadas e processadas

Informação

Dados

As intruções são as

Operações

que vão processar  
esses dados.

Instruções

## Etapa 2

# Instruções primitivas

// Primeiros passos para começar a programar/  
Fundamentos de algoritmos

Obs. usada para guardar valores, ela recebe um valo

[255]

# Instruções Primitivas

Cálculo matemáticos

Variáveis

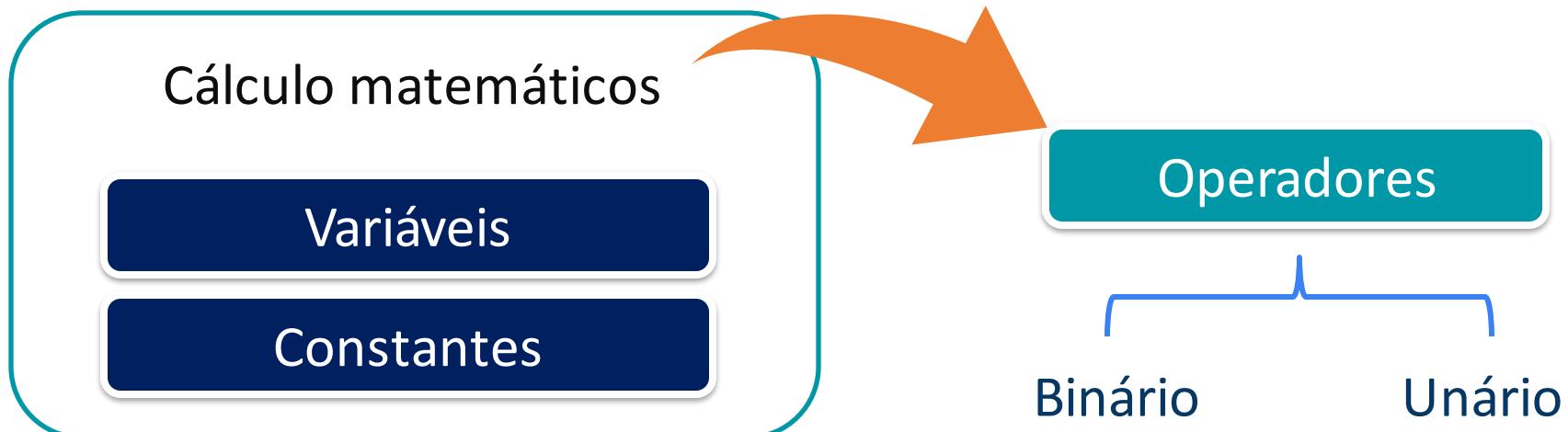
Constantes



Operadores

Obs. usada para guardar valores, ela recebe um valor, que tem que ser igual a [256]

# Instruções Primitivas



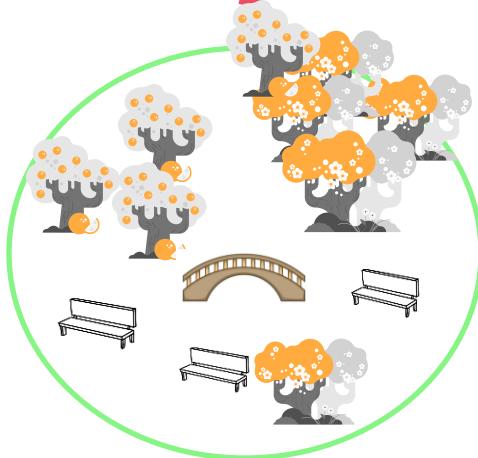
# Instruções Primitivas

## 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

fonte: livro de referência

# Instruções Primitivas



$$\text{Area} = \pi * \text{raio}^2$$

## 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

fonte: livro de referência

# 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?

# Instruções Primitivas

Linguagens de programação

Notação



Operações

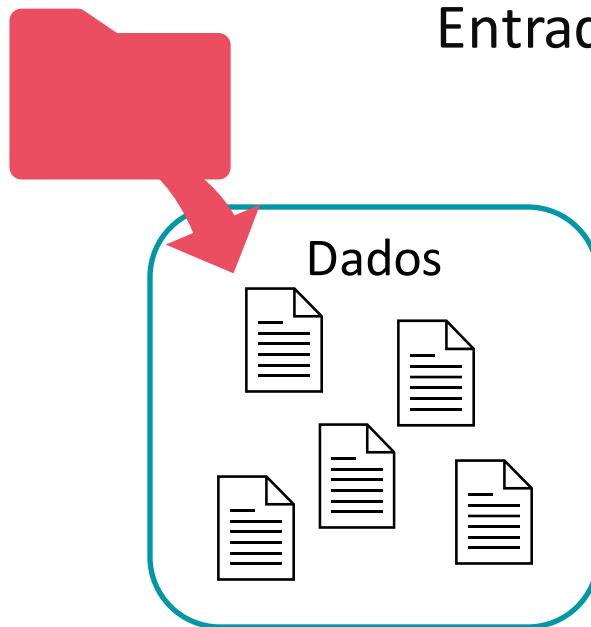
Janela

Window

Ventana

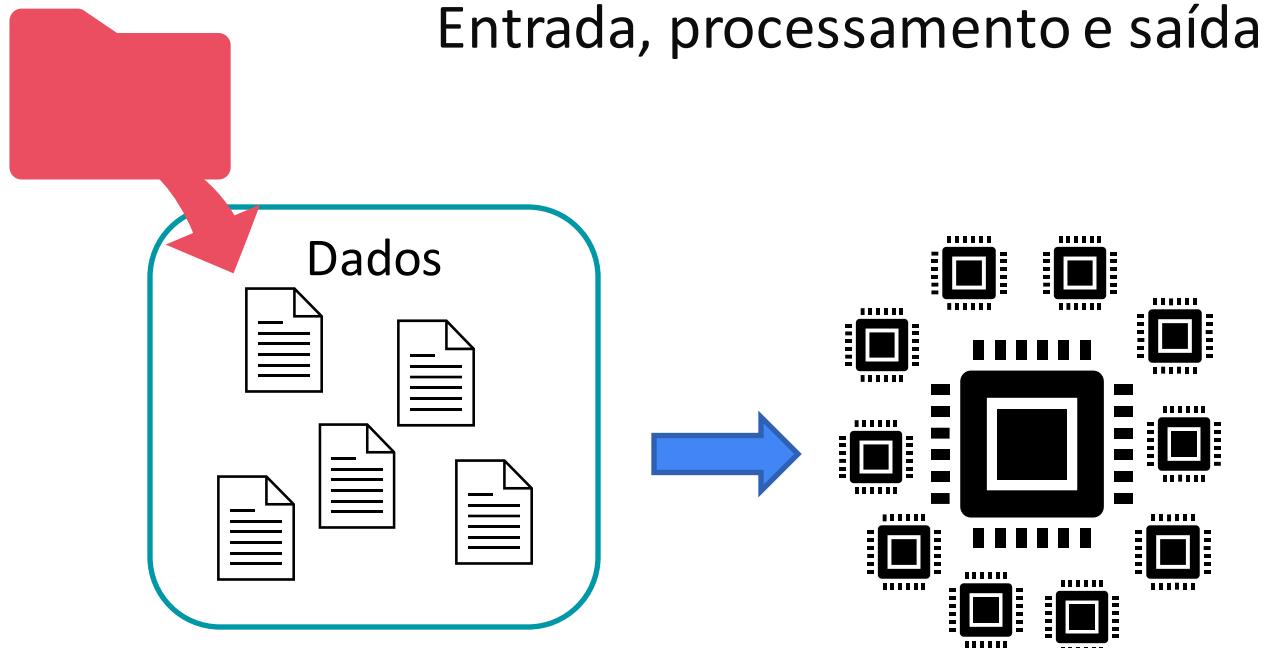


# Outros conceitos



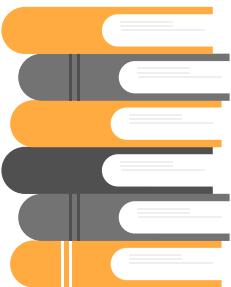
Entrada, processamento e saída

# Outros conceitos



# Exemplo

Média escolar



Início programa:

```
Nota1 = 5  
Nota = 8  
Resultado = 0
```

```
Resultado = (Nota1 + Nota2)/2
```

Escreva resultado

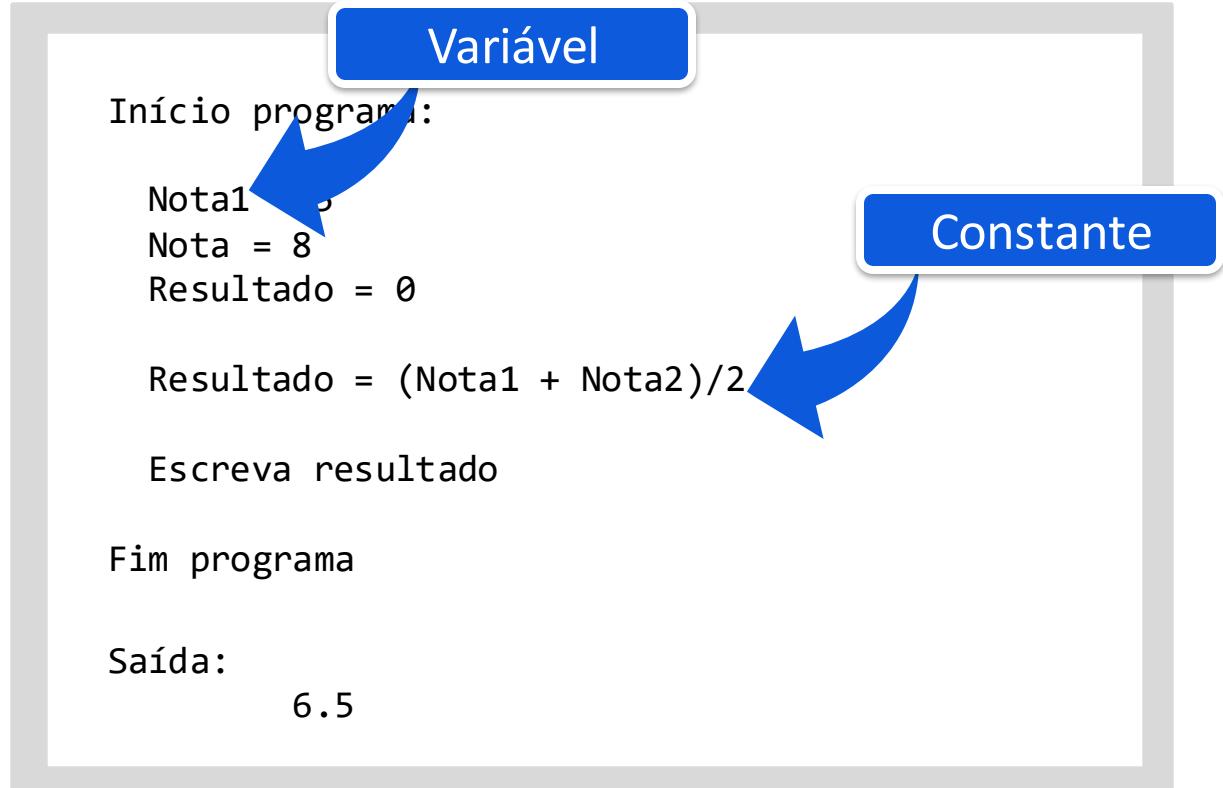
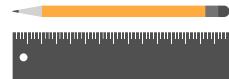
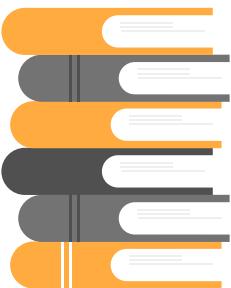
Fim programa

Saída:

6.5

# Exemplo

Média escolar



# Exemplo

Média escolar

Estrutura sequencial



Variável

Início programa:

Nota1

Nota = 8

Resultado = 0

Resultado = (Nota1 + Nota2)/2

Constante

Escreva resultado

Fim programa

Saída:

6.5

# Exemplo

Média escolar

E se quisermos verificar se foi  
aprovado ou não?



Variável

Início programa:

Nota1

Nota = 8

Resultado = 0

Constante

Nota = (Nota1 + Nota2)/2

resultado

Fim programa

Saída:

6.5

## Etapa 3

# Estruturas condicionais e operadores

// Primeiros passos para começar a programar/  
Fundamentos de algoritmos

# Exemplo

Média escolar

E se quisermos verificar se foi  
aprovado ou não?



Variável

Início programa:

Nota1

Nota = 8

Resultado = 0

Constante

Nota = (Nota1 + Nota2)/2

resultado

Fim programa

Saída:

6.5

# Estrutura Condicional

Estado de uma pessoa ou coisa



Condição

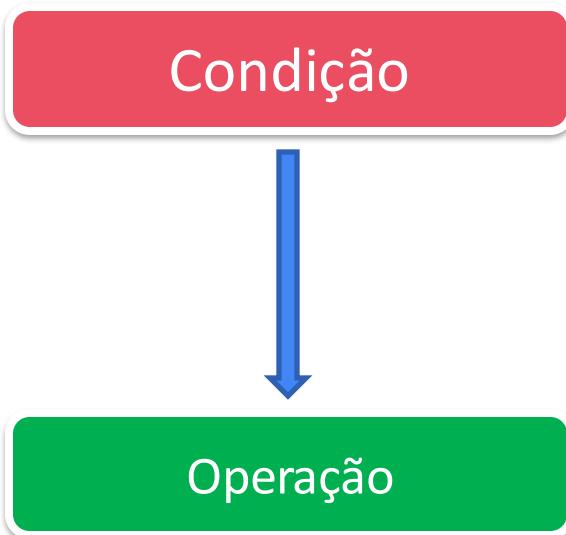
Condisional



Que expressão uma condição  
ou suposição

Contem ou implica uma suposição  
ou hipótese

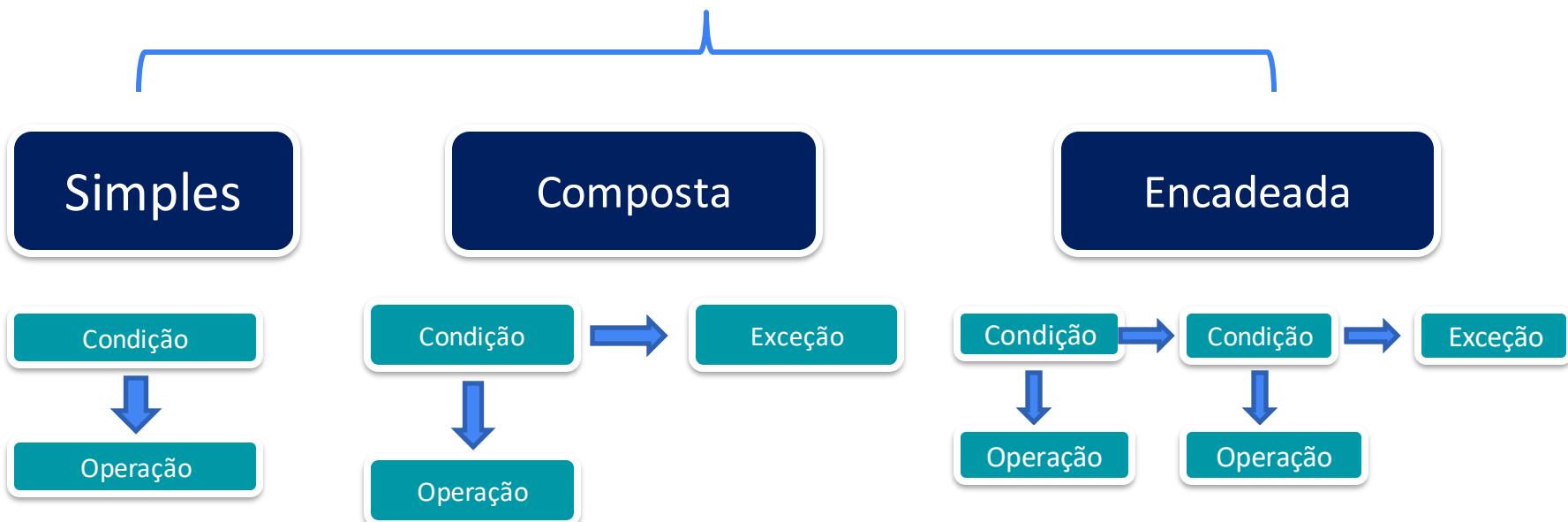
# Estrutura Condicional



# Estrutura Condicional



# Estrutura Condicional



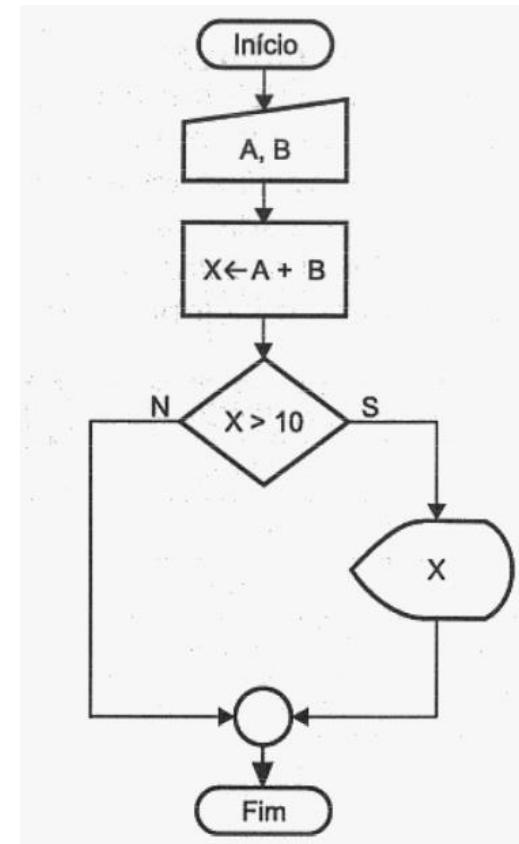
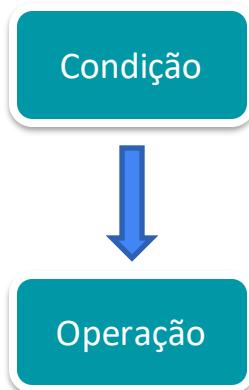
# Estrutura Condicional

Operadores Relacionais

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

# Estrutura Condicional

## Condisional Simples

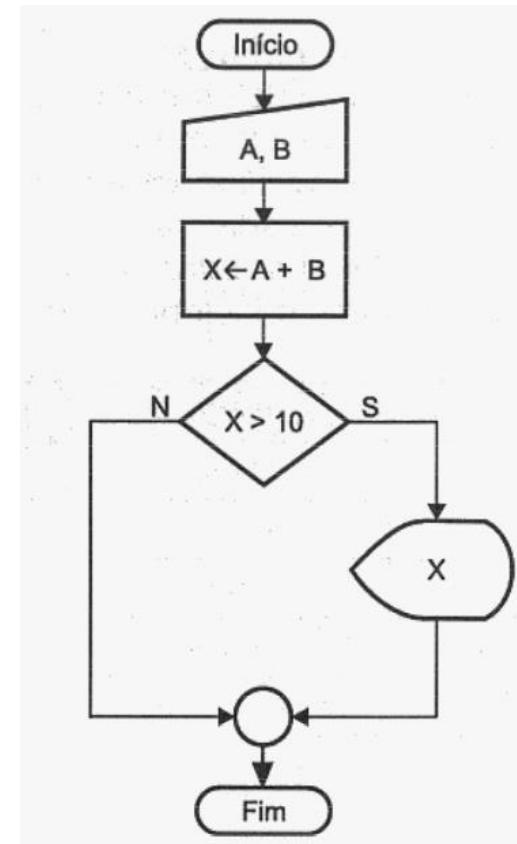


Fonte: livro de referência

# Estrutura Condicional

## Condisional Simples

Se (<condição>) então  
    <instruções para condição verdadeira>  
fim\_se



Fonte: livro de referência

# Estrutura Condicional

Início programa:

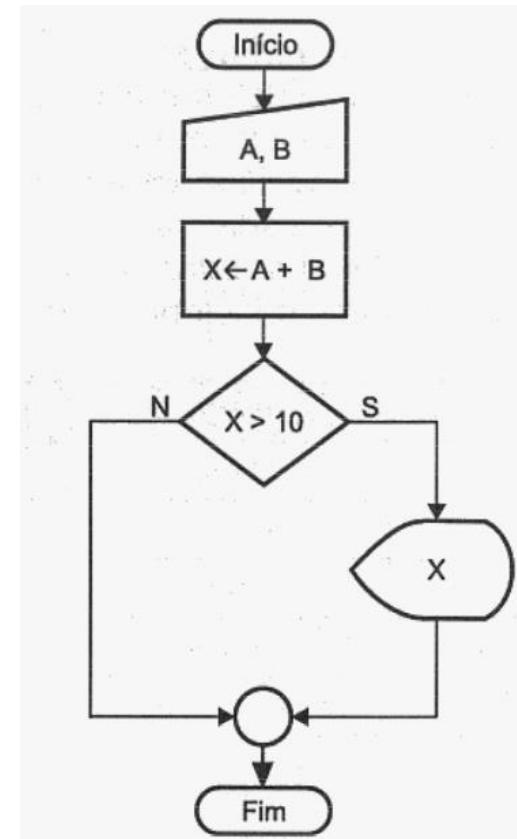
```
A = 0  
B = 0  
X = 0
```

```
leia A  
leia B
```

```
X = A + B
```

```
se (X > 10)  
    escreva X  
Fim se
```

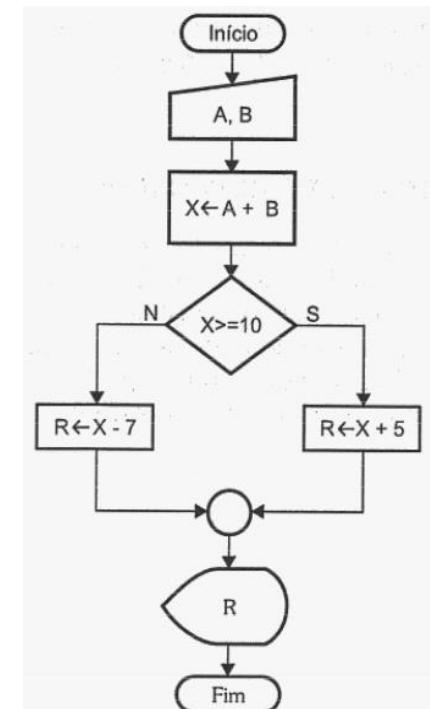
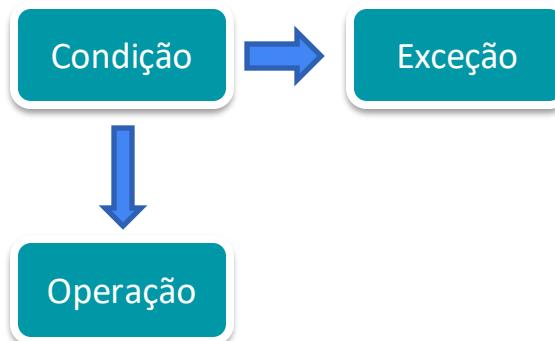
Fim programa



Fonte: livro de referência

# Estrutura Condicional

## Condisional Composta

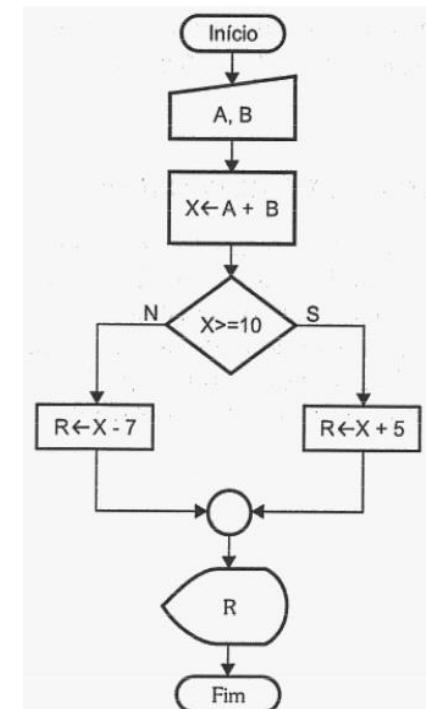


Fonte: livro de  
referência

# Estrutura Condicional

## Condisional Composta

Se (<codição>) então  
    <instruções para condição verdadeira>  
fim\_se



Fonte: livro de  
referência

# Estrutura Condicional

Inicio programa:

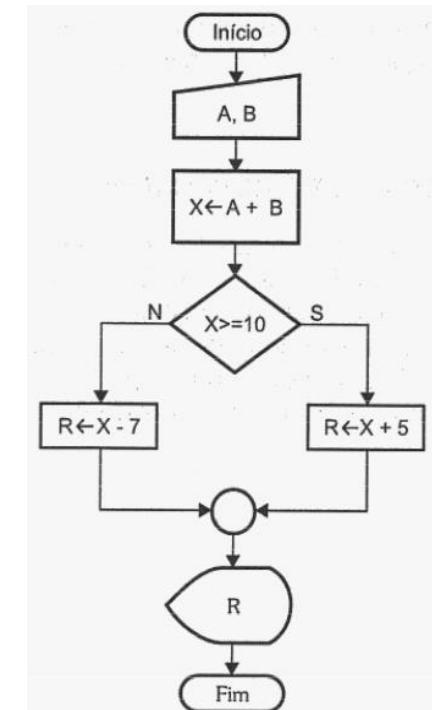
```
A = 0  
B = 0  
X = 0
```

```
leia A  
leia B
```

```
X = A + B
```

```
se (X > 10)  
    escreva X  
Fim se
```

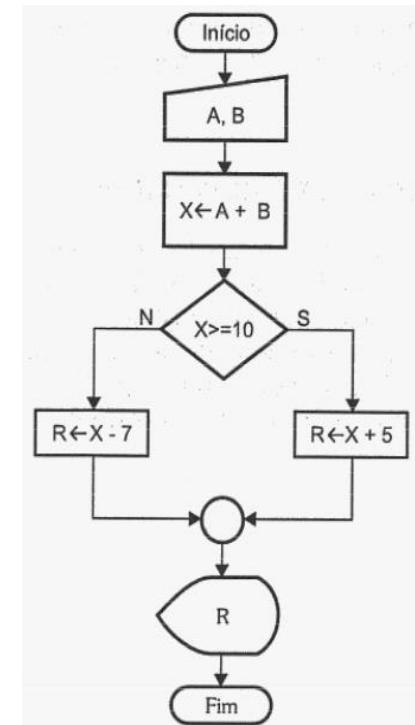
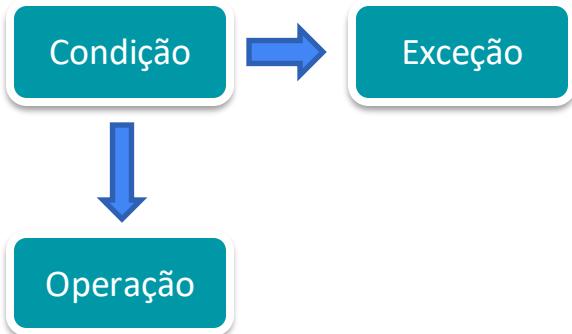
```
Fim programa
```



Fonte: livro de  
referência

# Estrutura Condicional

## Condisional Composta

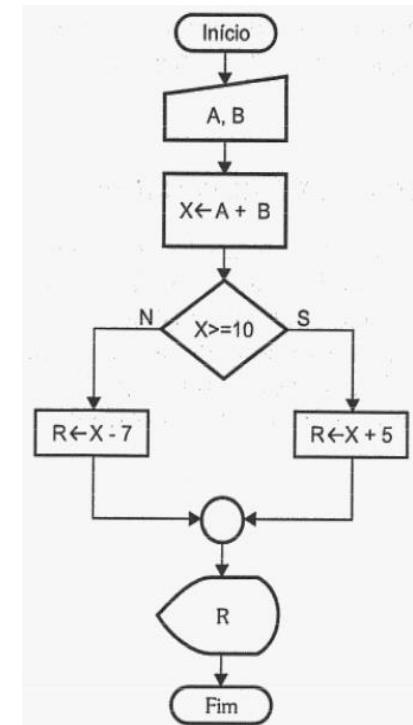


Fonte: livro de  
referência

# Estrutura Condicional

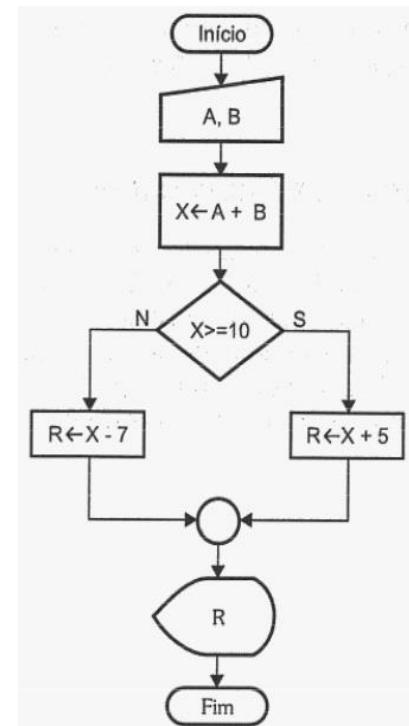
## Condisional Composta

Se (<codição>) então  
    <instruções para condição verdadeira>  
Senão  
    <instruções para condição falsa>  
fim\_se



Fonte: livro de  
referência

# Estrutura Condicional



Fonte: livro de  
referência

# Exemplo

Média escolar



Início programa:

```
Nota1 = 5  
Nota = 8  
Resultado = 0
```

```
Resultado = (Nota1 + Nota2)/2
```

Escreva resultado

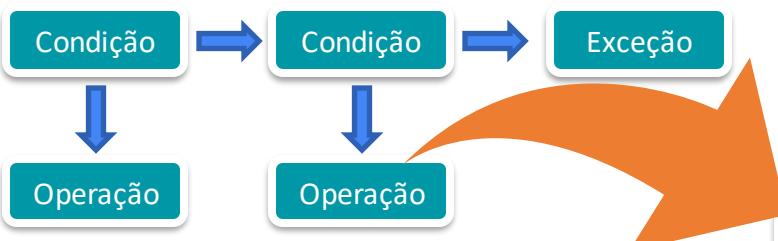
Fim programa

Saída:

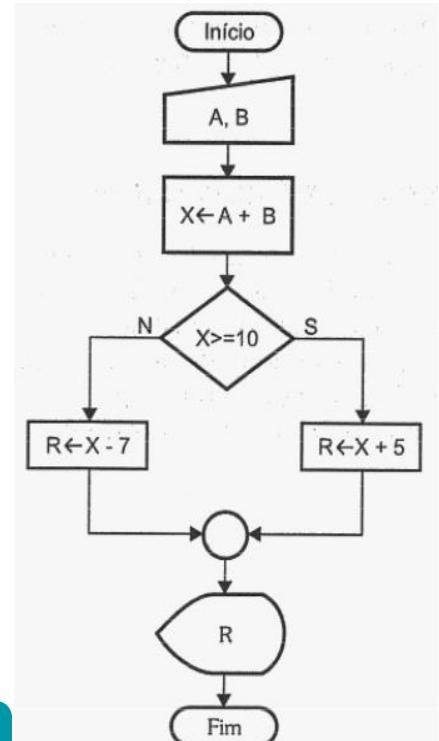
6.5

# Estrutura Condicional

## Condisional Encadeado



Condições sucessivas



Fonte: livro de  
referência

# Estrutura Condicional

## Condisional Encadeado

Se (<condição 1>) então  
    <instruções para condição verdadeira>

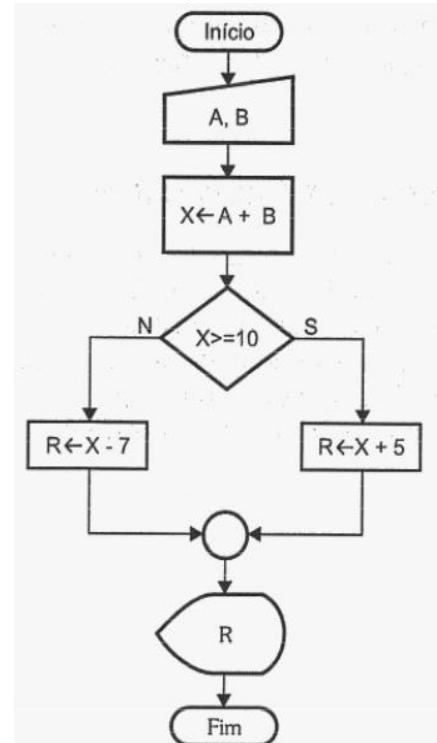
Senão

    Se (<condição 2>) então  
        <instruções para: condição 2 verdadeira e condição 1 falsa>

    Senão

        <instruções para condição 1 e 2 falsas>

fim\_se



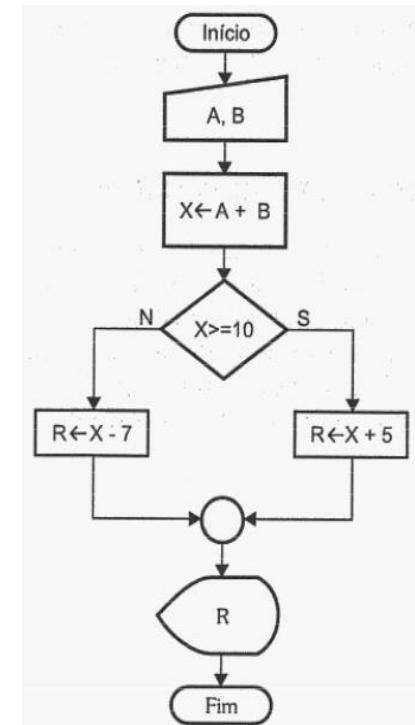
Fonte: livro de  
referência

# Estrutura Condicional

Início programa:

```
A = 0  
B = 0  
X = 0  
  
leia A  
leia B  
X = A + B  
se (X >= 10)  
    X = X-7  
Senão  
    X = X+5  
Fim se
```

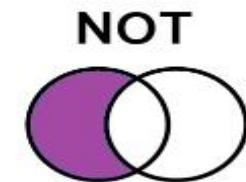
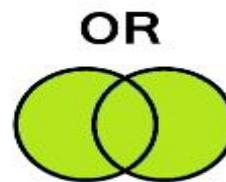
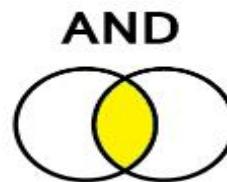
Fim programa



Fonte: livro de  
referência

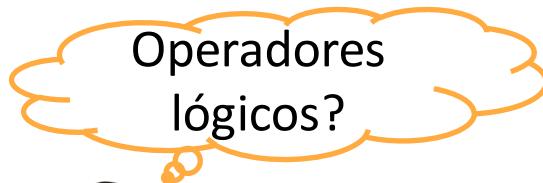
# Estrutura Condicional

Operadores Lógicos



Fonte: [biblio.info](http://biblio.info)

# Estrutura Condicional



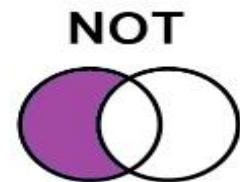
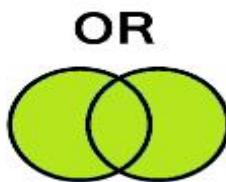
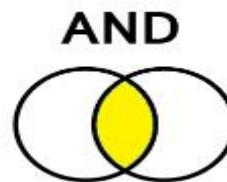
Quando utilizar?

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

# Estrutura Condicional

Quando utilizar?

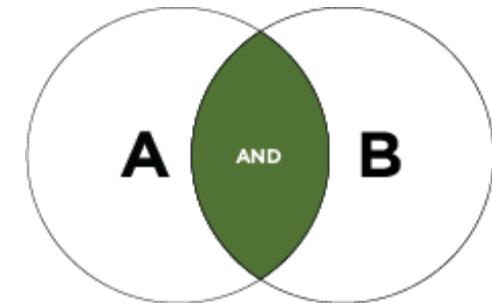
- Verificação de V ou F
- Substituição
  - encadeamento de condições



Fonte: [biblio.info](http://biblio.info)

# Estrutura Condicional

AND – operador Lógico



Fonte: dot.li.com

Condição verdadeira

- Todas devem ser satisfeitas

Condição 1	Condição 2	Resultado
Falsa	Falsa	Falso
Verdadeira	Falsa	Falso
Falsa	Verdadeira	Falso
Verdadeira	Verdadeira	Verdadeiro

Fonte: livro de referência

# Estrutura Condicional

AND – operador Lógico

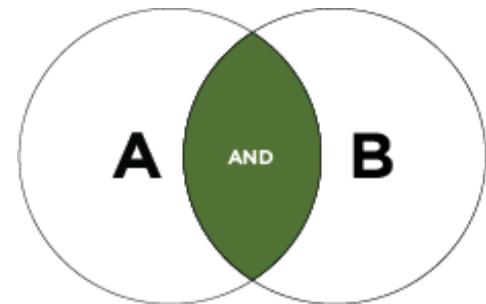
Ex: Curso de inglês

se (gramatica .e .conversacao) então

escreva "aprovado"

senão

escreva "Reprovado"



Fonte: dot.li.com

Interseção

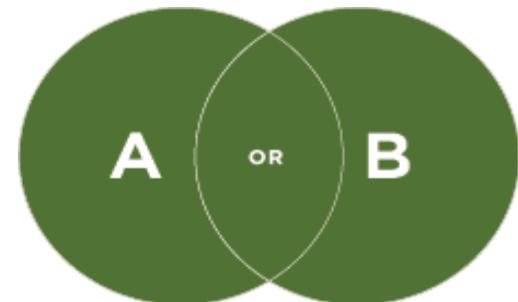
# Estrutura Condicional

OR – operador Lógico

Condição verdadeira

- Apenas uma das condições

Deve ser verdadeira



Fonte: dot.li.com

Condição 1	Condição 2	Resultado
Falsa	Falsa	Falso
Verdadeira	Falsa	Verdadeiro
Falsa	Verdadeira	Verdadeiro
Verdadeira	Verdadeira	Verdadeiro

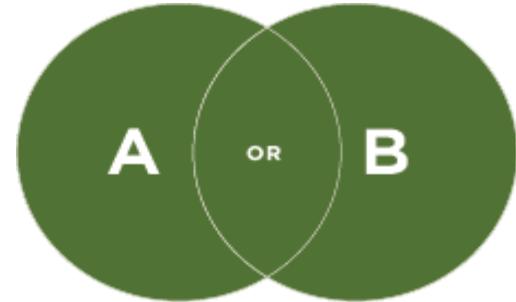
Fonte: livro de referência

# Estrutura Condicional

AND – operador Lógico

Ex:

```
se (choveu .ou. grama_molhada) então  
    escreva "Plantas regadas"  
senão  
    escreva "Regar plantas"
```



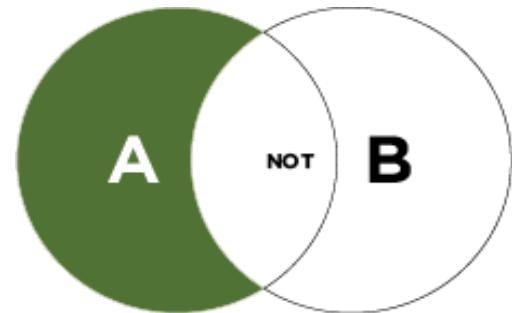
Fonte: dot.li.com



# Estrutura Condicional

NOT – operador Lógico

- Operador de negação
- Inversão do resultado lógico



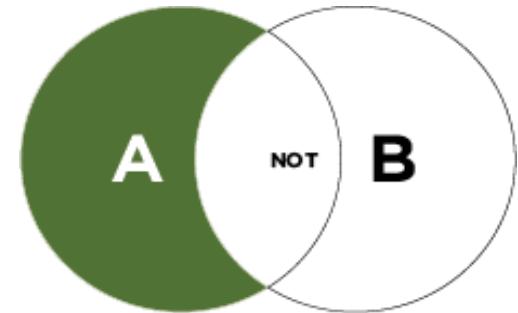
Condição	Resultado
Verdadeira	Falso
Falso	Verdadeira

Fonte: livro de referência

# Estrutura Condicional

NOT – operador Lógico

- Operador de negação
- Inversão do resultado lógico



Ex:

- Not B - > tudo que não está em B

## Etapa 4

# Estruturas de repetição

// Primeiros passos para começar a programar/  
Fundamentos de algoritmos

# Estruturas de Repetição



Laços

Controle de fluxo

Malhas de repetição

Repetição

Loop

# Estruturas de Repetição



Trecho de um  
programa

Condições de parada

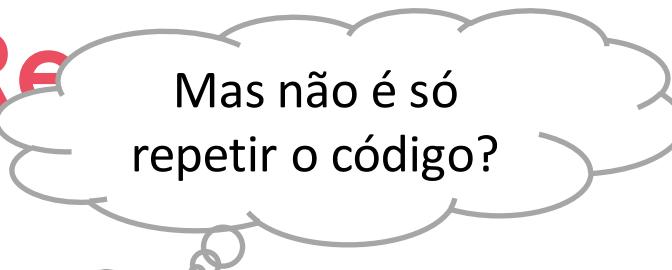
# Estruturas de Repetição



## Condições de parada

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

# Estruturas de Repetição



# Estruturas de Repetição

Trecho de um programa

Mas não é só repetir o código?

Qual a vantagem?

Redução de linhas

Compreensão facilitada

Redução de erro



# Estruturas de Repetição

Enquanto ... faça

Repita ... até

Para ... de ... até ... faça

Tem mais de tipo  
de estrutura de  
repetição?



# Estruturas de Repetição

Enquanto

Teste lógico

- Início

Número de repetições

- indefinidas

Grama alta



Fonte: pt.vecteezy.com

# Estruturas de Repetição

Enquanto

Teste lógico

- Início

Número de repetições

- indefinidas

Grama alta

Aparar grama



Fonte: pt.vecteezy.com

# Estruturas de Repetição

Enquanto

Teste lógico

- Início

Número de repetições

- indefinidas

Grama alta

Analisar grama

Aparar grama



Fonte: pt.vecteezy.com

# Estruturas de Repetição

Enquanto

Teste lógico

- Início

Número de repetições

- indefinidas

grama = Falso

Enquanto (grama == falso) faça

<instrução de cortar grama>  
<atualiza grama>

fim enquanto

# Estruturas de Repetição

Repita

Teste lógico

- final

Número de repetições

- indefinidas

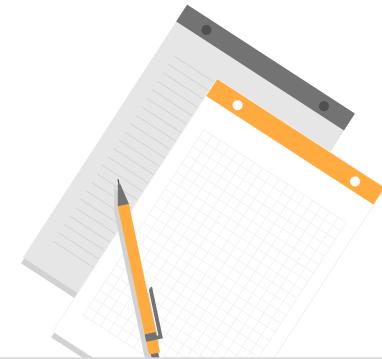
Procurar artigo

Virar página

Analisar conteúdo



# Estruturas de Repetição



Para ... de ... até

Teste

- Início

Número de repetições

- Definidas

```
somatorio = 0
```

```
para inicio = 1 até 10
```

```
    somatorio = somatorio + inicio  
    Escreva somatorio
```

```
fim para
```

# Estruturas de Re



Consigo mesclar  
as estruturas?

Uma dentro  
da outra?

# Estruturas de Re



Trecho de um  
programa



Consigo mesclar  
as estruturas?

Uma dentro  
da outra?



Enquanto (<condição>)

Se (<condição2>)  
<instruções>  
fimse

...

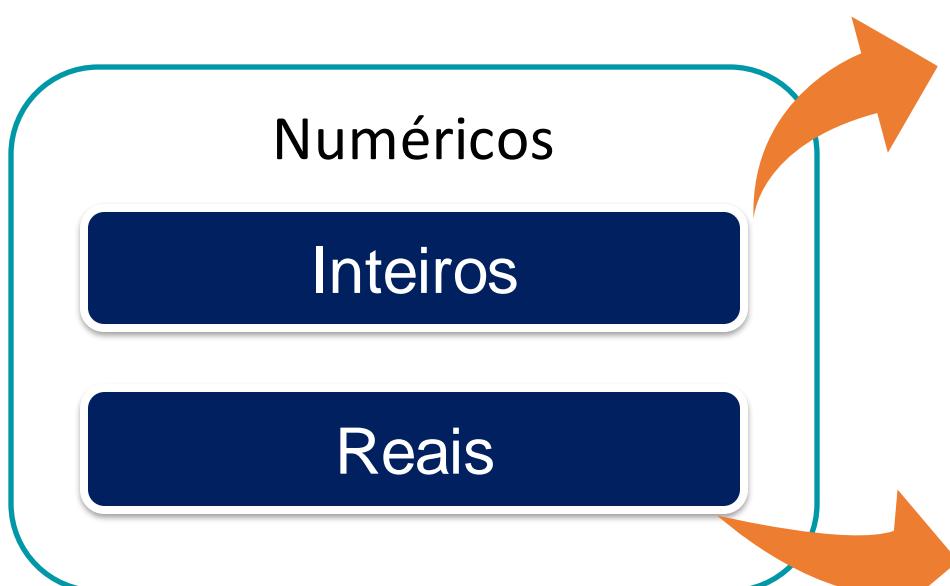
fim enquanto

## Etapa 5

# Vetores e matrizes

// Primeiros passos para começar a programar/  
Fundamentos de algoritmos

# Vetores e Matrizes



0, 1, 5, 50, 60 , 800, ...  
-58, -50, -49, 32, -10, -5

5.95, 9.54, -8.8, -0.555 ...  
0, 1, 5, 50, 60 , 800, ...  
-58, -50, -49, 32, -10, -5

# Vetores e Matrizes

Container

Definição formal

"Um vetores é caracterizado por uma variavel dimensionada com tamanho pré-fixado."

Matriz unidimensional

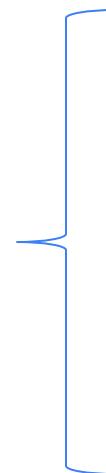
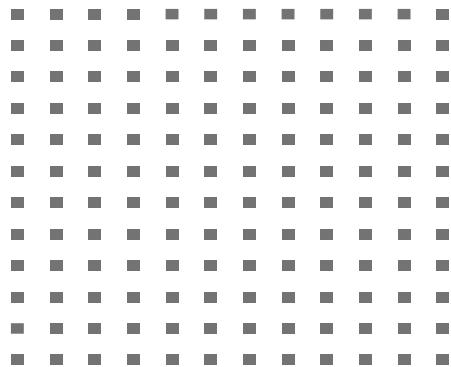
# Vetores e Matrizes

Container

## Definição formal

"Matriz é uma tabela organizada em linhas e colunas no formato  $m \times n$ , onde  $m$  representa o número de linhas (horizontal) e  $n$  o número de colunas (vertical)"

# Vetores e Matrizes



Coleção de variáveis

Contiguas em memória

Índices

Matrizes

# Vetores e Matrizes

## Média escolar



```
nota11 = 10  
nota21 = 5
```

```
nota12 = 7  
nota22 = 8
```

---

```
notas_aluno1 = [10,5]  
notas_aluno2 = [7,8]
```

# Vetores e Matrizes

## Média escolar



```
nota11 = 10  
nota21 = 5
```

```
nota12 = 7  
nota22 = 8
```

---

```
notas_aluno1 = [10,5]  
notas_aluno2 = [7,8]
```

---

```
notas_alunos = [10,5,7,8]
```

A cada duas posições  
um novo aluno



# Vetores e Matrizes

## Definição

vetor conjunto [1..8]<inteiro>

vetor[15]

Vetor = []

## Atribuição

## Leitura

## Operações



# Vetores e Matrizes

Aluno	Nota 1	Nota 2	Nota 3	Nota 4	Média
1	4.0	6.0	5.0	3.0	4.5
2	6.0	7.0	5.0	8.0	6.5
3	9.0	8.0	9.0	6.0	8.0
4	3.0	5.0	4.0	2.0	3.5
5	4.0	6.0	6.0	8.0	6.0
6	7.0	7.0	7.0	7.0	7.0
7	8.0	7.0	6.0	5.0	6.5
8	6.0	7.0	2.0	9.0	6.0

Dados de alunos

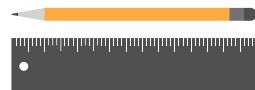
# Vetores e Matrizes

Aluno	Nota 1	Nota 2	Nota 3	Nota 4	Média
1	4.0	6.0	5.0	3.0	4.5
2	6.0	7.0	5.0	8.0	6.5
3	9.0	8.0	9.0	6.0	8.0
4	3.0	5.0	4.0	2.0	3.5
5	4.0	6.0	6.0	8.0	6.0
6	7.0	7.0	7.0	7.0	7.0
7	8.0	7.0	6.0	5.0	6.5
8	6.0	7.0	2.0	9.0	6.0

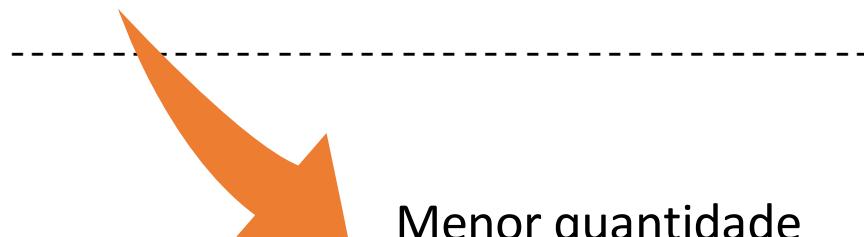
Dados de alunos

# Vetores e Matrizes

Média escolar



```
matriz_alunos[6][6]  
  
escreva "ID aluno"  
escreva matriz_alunos[0][0]  
escreva "Média"  
escreva matriz_alunos[0][6]
```



Menor quantidade  
de linhas

## Etapa 6

# O que são funções?

// Primeiros passos para começar a programar/  
Fundamentos de algoritmos

# Funções

Subalgoritmo

Subprograma

Função

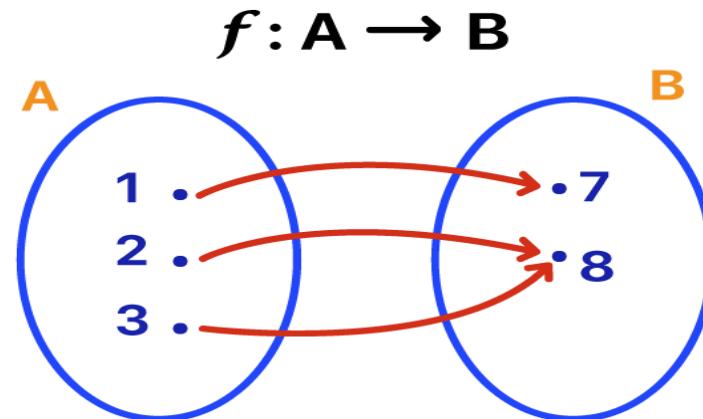
Bloco

Método

Sub-rotina

# Funções

Similar ao conceito de função matemática



[blog.professorferretto.com.br](http://blog.professorferretto.com.br)

# Funções

Definição formal

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

Decomposição do algoritmo

# Funções

As funções, como os métodos, são blocos de instruções que realizam tarefas específicas

Definição  
Modularização do problema

Decomposição do algoritmo

# Funções

Função? Não é mais fácil fazer tudo junto?



Modularização do programa

Código mais claro e conciso

Reutilização de instruções

# Funções

Função? Não é mais fácil  
fazer tudo junto?



Definição formal

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

Assinatura

# Funções



Definição formal

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

# Funções

Definição

Nome

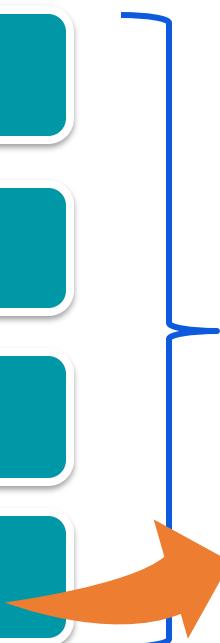
Invocação

Variável local

Definição formal

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

São destruídas ao encerrar a função



# Funções



# Funções

Alterar estado do programa



# Funções

Agora vamos determinar a média escolar através de uma função.

Qual o ganho que teremos ao fazer essa modificação?

Exemplo

# Funções

Reutilização de código

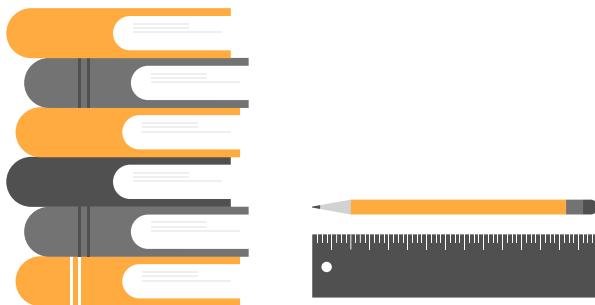
Agora vamos determinar a média escolar através de uma função.

Qual o ganho que teremos ao fazer essa modificação?

Exemplo

# Funções

## Média escolar



```
funcao mediaescolar(nota1,nota2)
```

```
    resultado = 0
```

```
    resultado = (Nota1 + Nota2)/2
```

```
    retorno resultado
```

```
fim funcao
```

---

```
aluno1 = mediaescolar(nota11,nota21)
```

```
aluno2 = mediaescolar(nota12,nota22)
```

```
aluno13 = mediaescolar(nota13,nota23)
```

## Etapa 7

# Instruções de entrada/saída

// Primeiros passos para começar a programar/  
Fundamentos de algoritmos

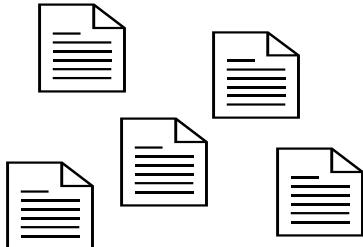
# Instruções de entrada/saída



# Instruções de entrada/saída

Quais os tipos e como inserir os dados?

Dados



Como exibir meu resultado?

Resultado

Processamento

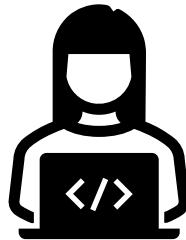
# Instruções de entrada/saída

Definição formal

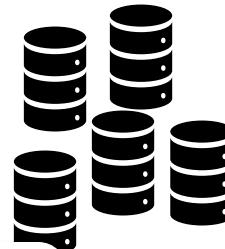
Consiste na inserção e recebimento de dados do mundo real por meio de ação de alguma interface, seja teclado, mouse, arquivos, entre outros.

Instruções de entrada

# Instruções de entrada/saída

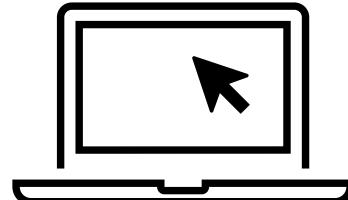
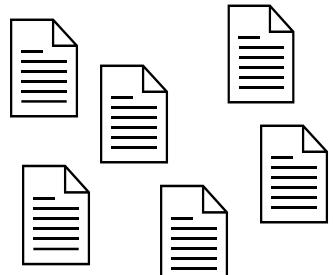


Metadados



Inserção de dados

Arquivos



# 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

# Instruções de entrada/saída



Arquivos



Dispositivos de saída



Log do sistema

Arquivos de acesso  
ao usuário

Híbridos



# Instruções de entrada/saída

Existem dois tipos de saídas dentro de um programa (algoritmo)

Saída programada

Saída por interrupção

Definida pelos periféricos



# Instruções de entrada/saída

Existem dois tipos de saídas dentro de um programa (algoritmo)

Saída programada

Saída por interrupção

Condicional

Incondicional



# Instruções de entrada/saída

Existem dois tipos de saídas dentro de um programa

Aguarda o dispositivo



Condicional

Incondicional

Saída programada

Saída por interrupção



# Instruções de entrada/saída

Casos

- Bem sucedida
- Erro de sintaxe ou outro
- Erros de programação
- Problemas com a interface

Saída na programação



código

# Aula 4: Linguagens de programação

// Primeiros passos para começar a programar

# Objetivo

O objetivo desta aula é apresentar os paradigmas de programação existentes, incluindo o conceito em si. As linguagens de programação são baseadas em um ou mais paradigmas, sendo o último caso conhecido como linguagem multiparadigma.

# Percorso

Etapa 1

Introdução à linguagem de programação

Etapa 2

Como um computador entende o programa?

Etapa 3

Características de um programa

# Percurso

**Etapa 4** Análises e código

**Etapa 5** Paradigmas de programação

## Etapa 1

# Introdução à linguagem de programação

// Primeiros passos para começar a programar/  
Linguagens de programação

# Introdução

Por que falar de história da computação?

- Compreender as dificuldades enfrentadas; que são as mesmas de hoje
- Fundamentos da computação;
- O processo de pensamento.

# Introdução

Por que falar de história da computação?

- Compreender as dificuldades enfrentadas;
- Fundamentos da computação;
- O processo de pensamento.

Base em pesquisas anteriores

Novos paradigmas



# Introdução

Por que falar de história da computação?

- Compreender as dificuldades enfrentadas;
- Fundamentos da computação;
- O processo de pensamento.

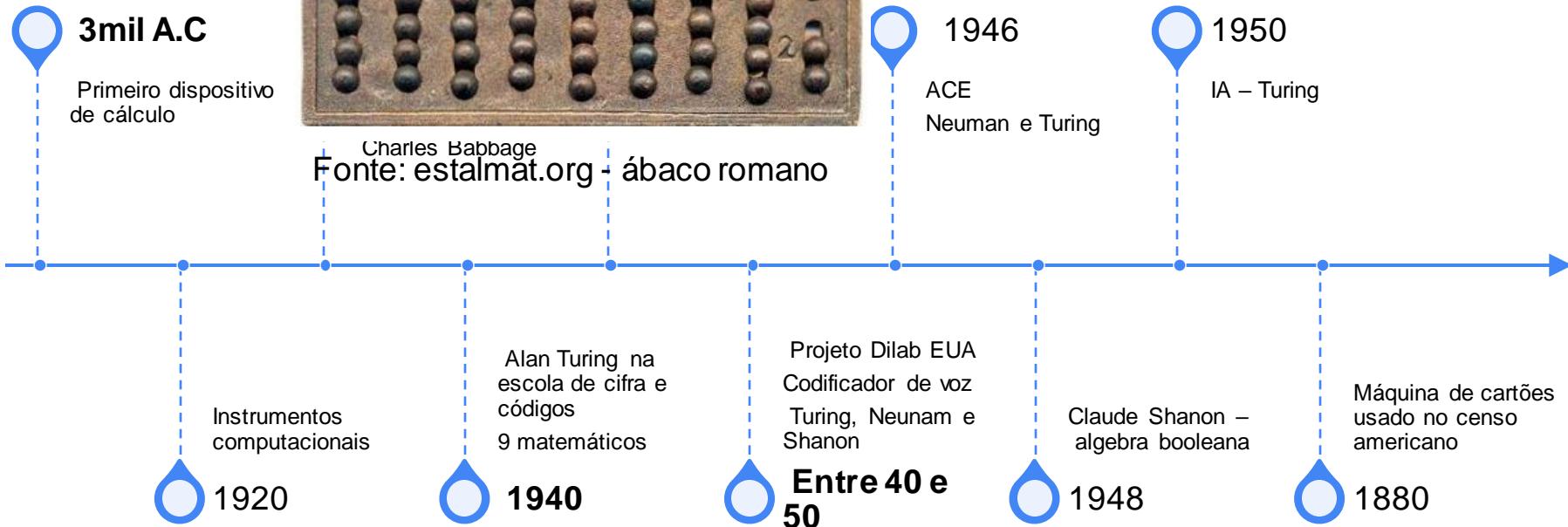
Base em pesquisas anteriores

Novos paradigmas



# Introdução

## História da computação



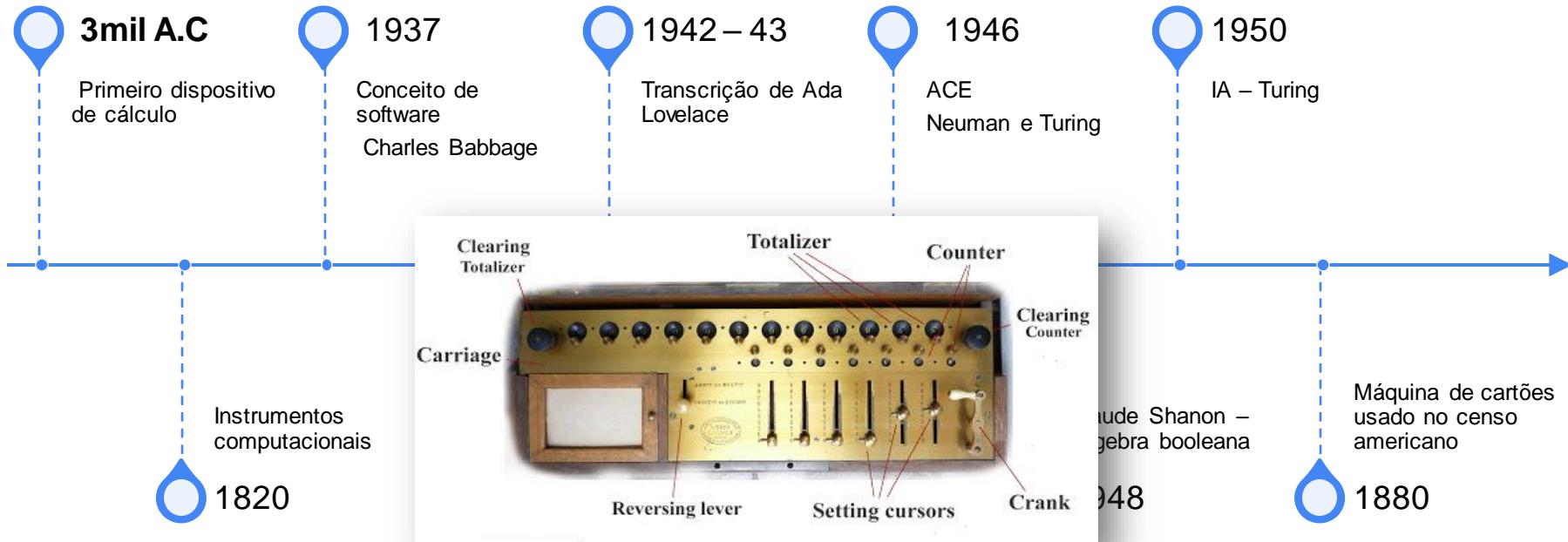
# Introdução

## História da computação



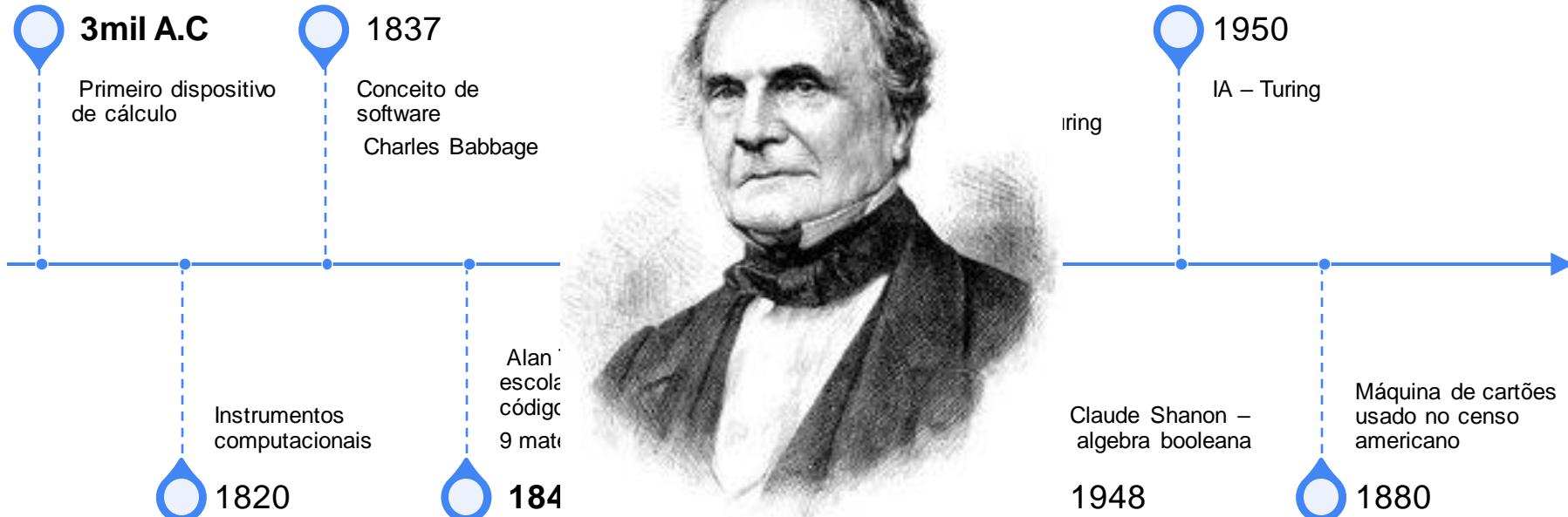
# Introdução

## História da computação

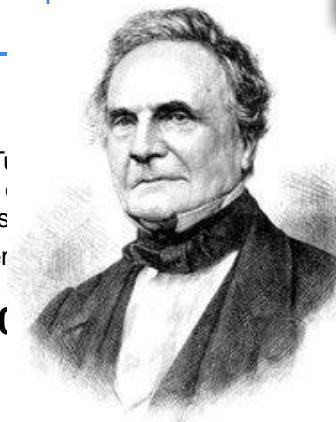
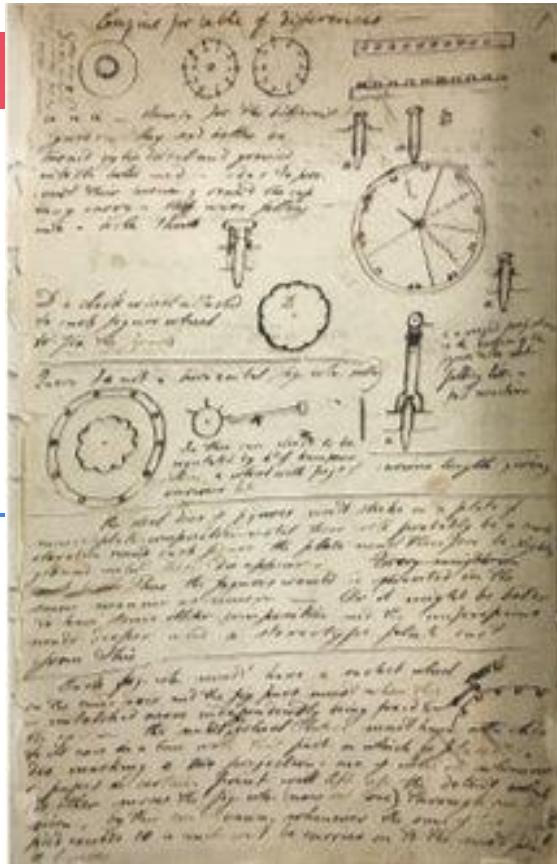


# Introdução

## História da computação



## História da computação



1942 – 43

Transcrição de Ada Lovelace



40 e

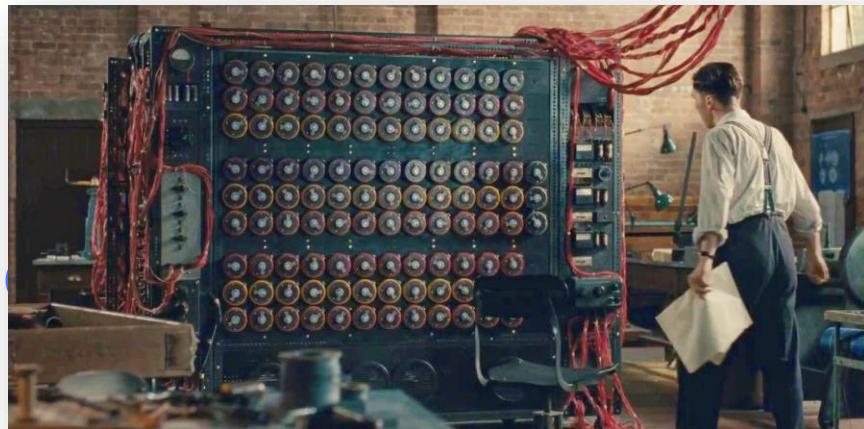
1948

silab EUA  
or de voz  
leunam e

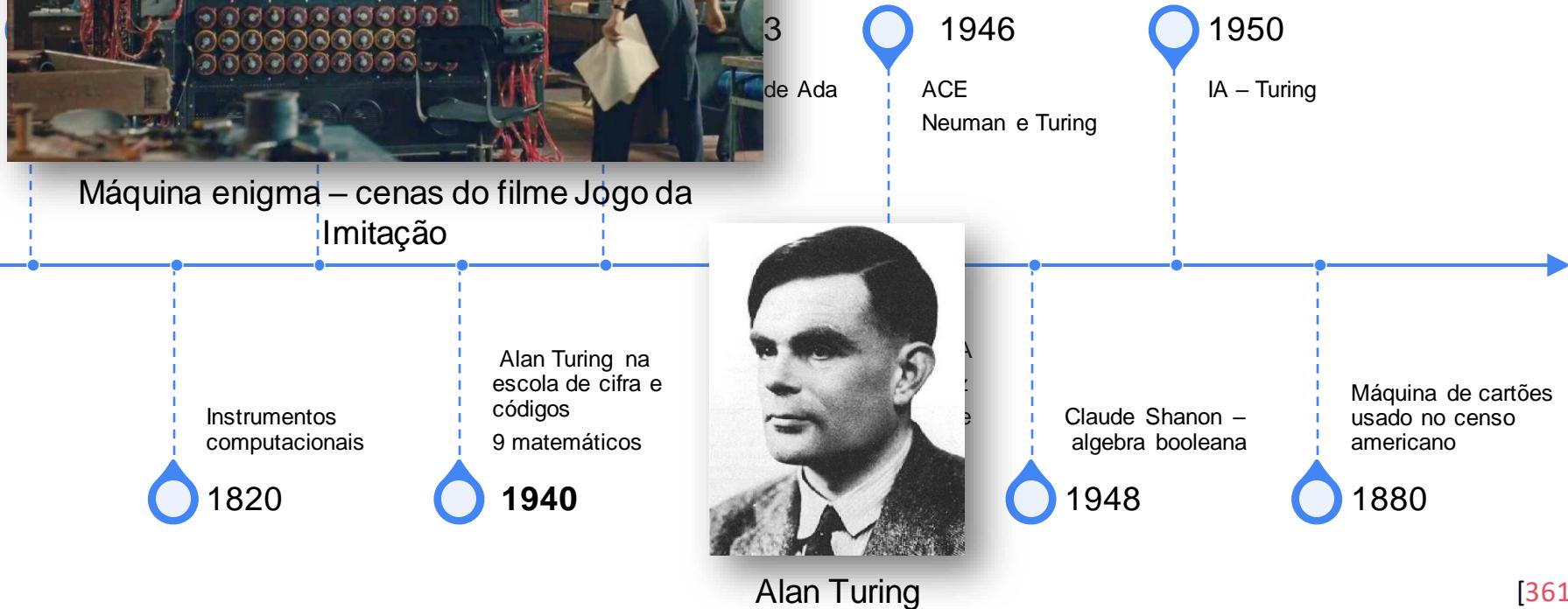
Claude Shannon –  
álgebra booleana

1880

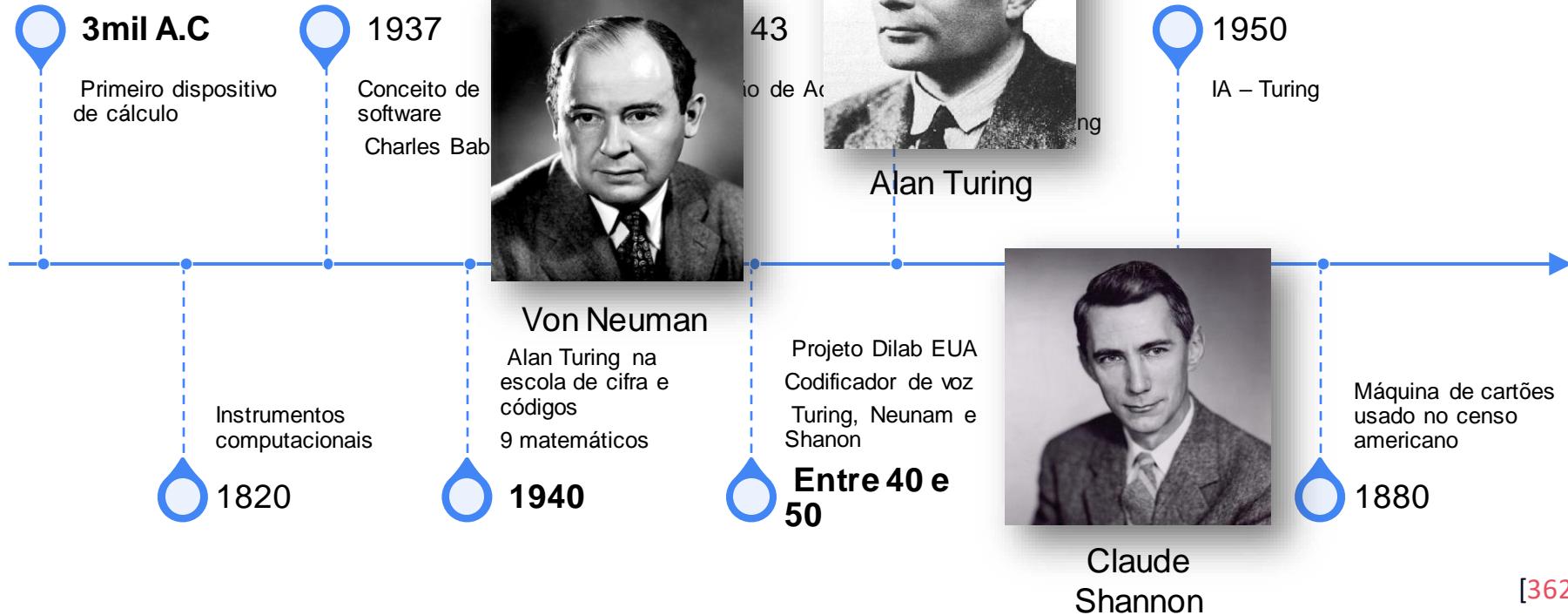
Máquina de cartões  
usado no censo  
americano



## História da computação



# Introdução

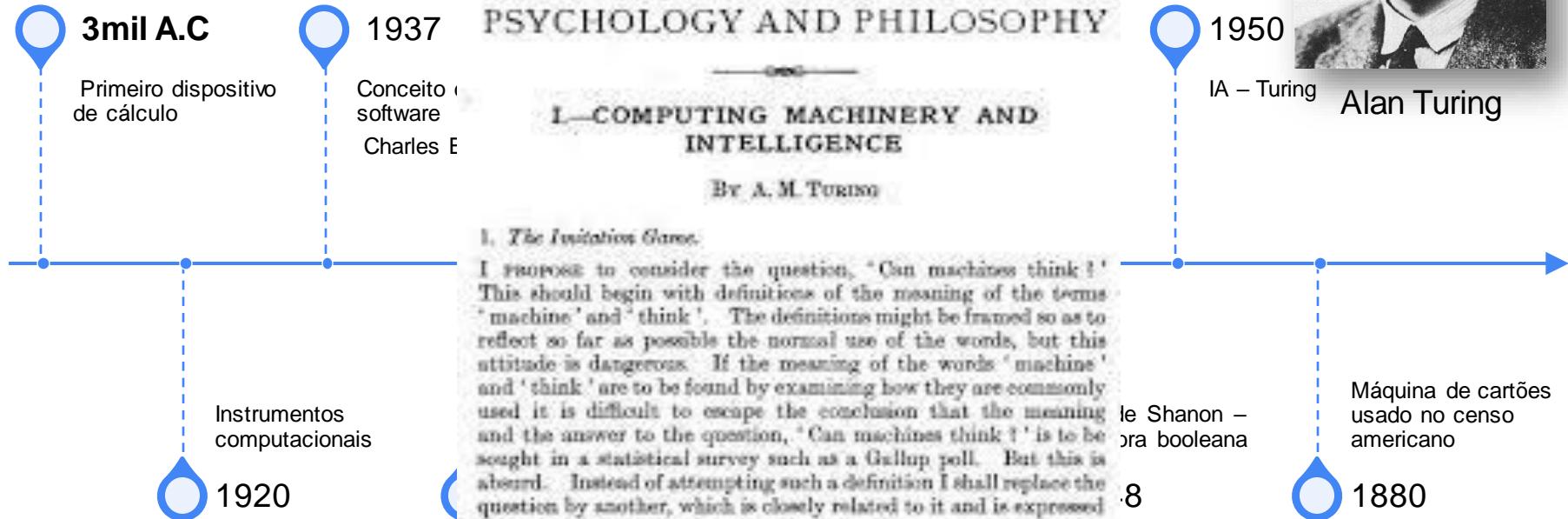


# Introdução

## História da computação



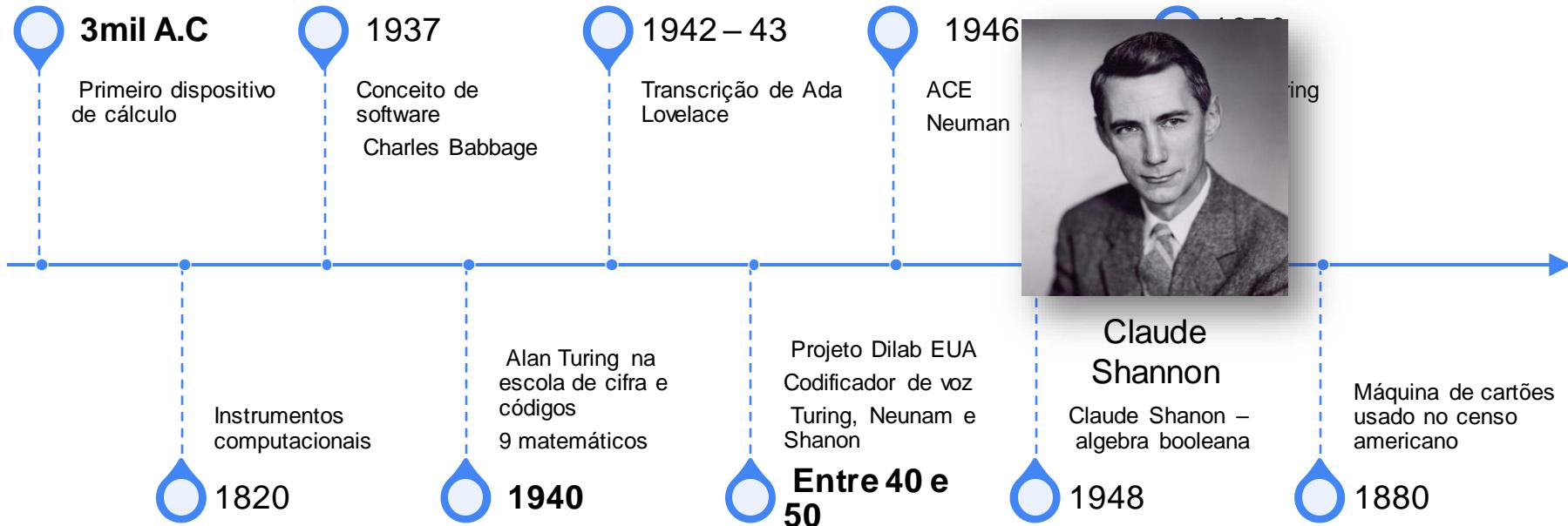
# Introdução



Fonte: jstor.org

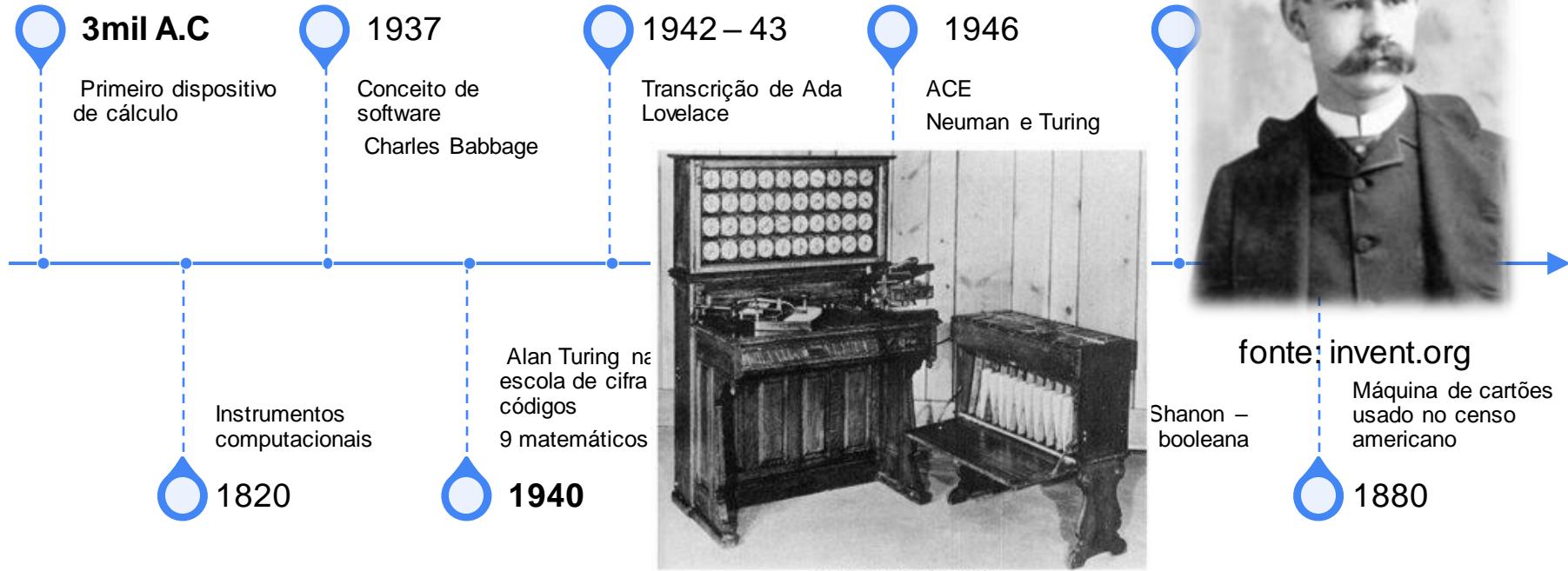
# Introdução

## História da computação



# Introdução

## História da com



# Introdução

## História da com

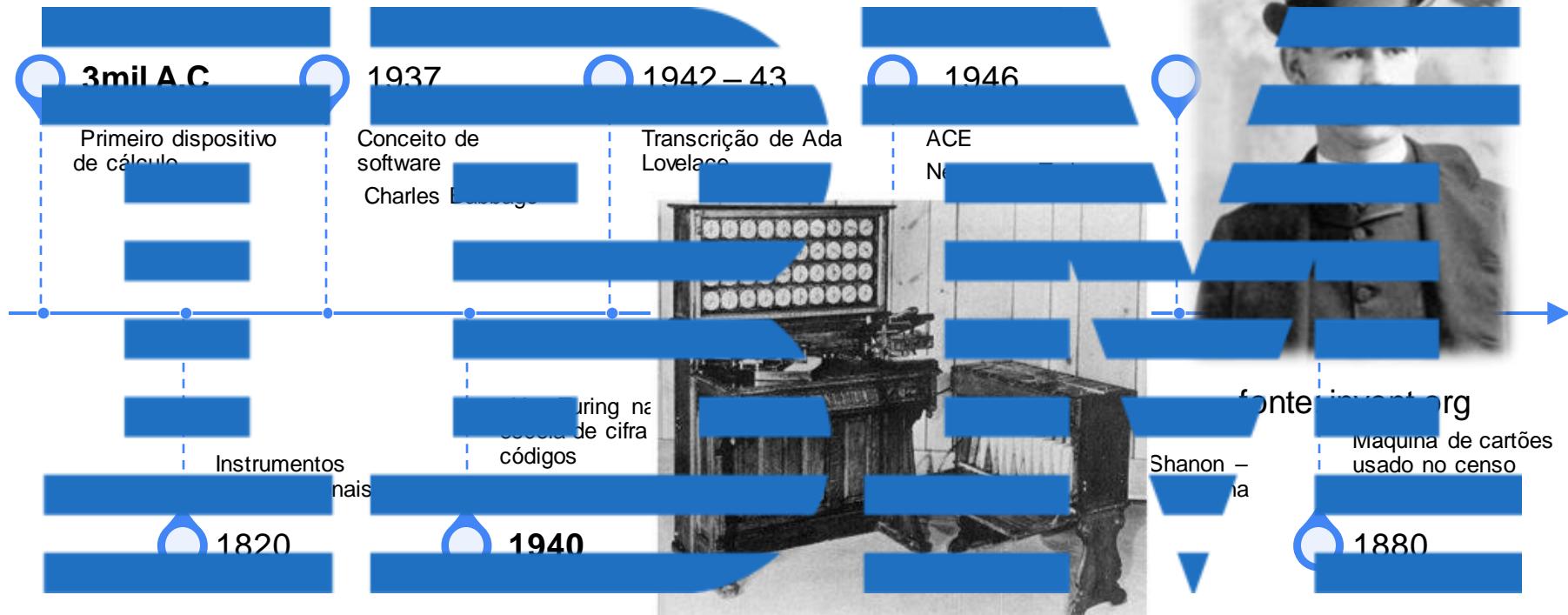
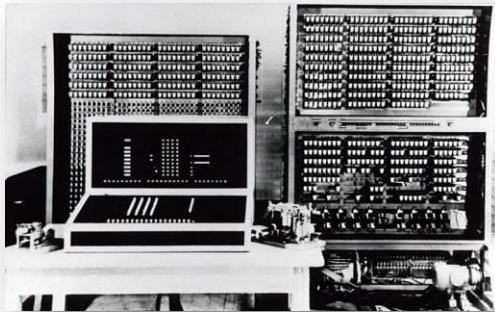


Figura 25: Tabuladora de Hollerith



Fonte: inf.ufrgs.br

ão

2° guerra mundial



## Primeiros computadores

Válvula

Relés - avanço



## ENIAC

Programável

18 mil válvulas

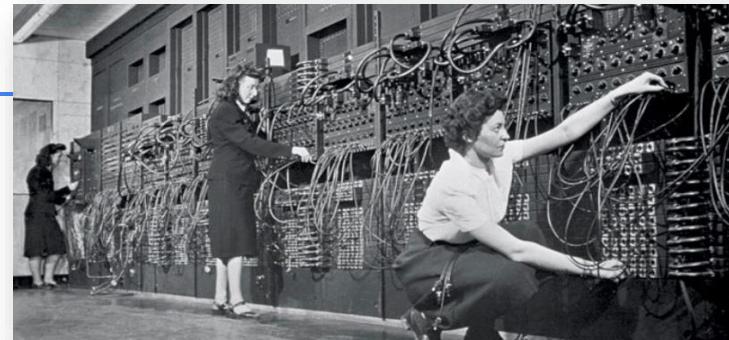


IBM

Separação de memórias

Dcisão - algoritmo

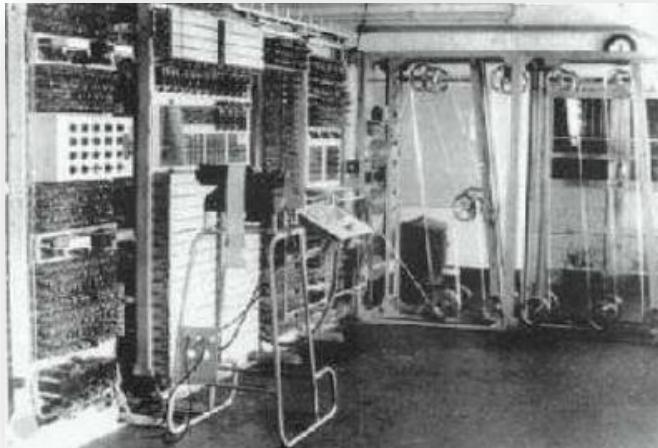
Harvard Mark 1



Mulheres programando no ENIAC

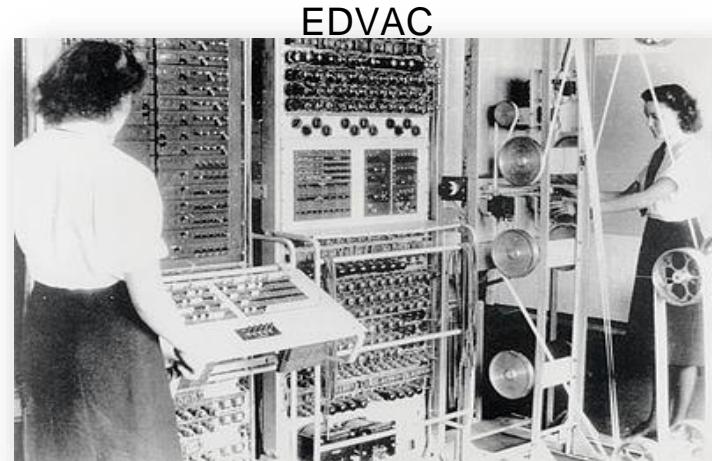
# Introdução

# 1943



Colossus - programação por fio

# 1952



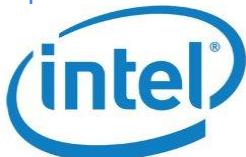
Programa armazenado

# Intro

1975  
Intel 8080  
Linguagem  
basic -  
Microsoft



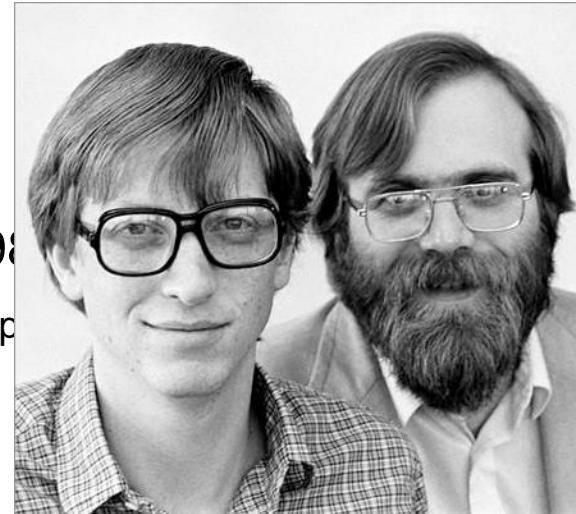
fonte:  
[computerhistory.org](http://computerhistory.org)



App  
1971



Microsoft



1981

App

[:wired.com](http://wired.com)

acasso  
»

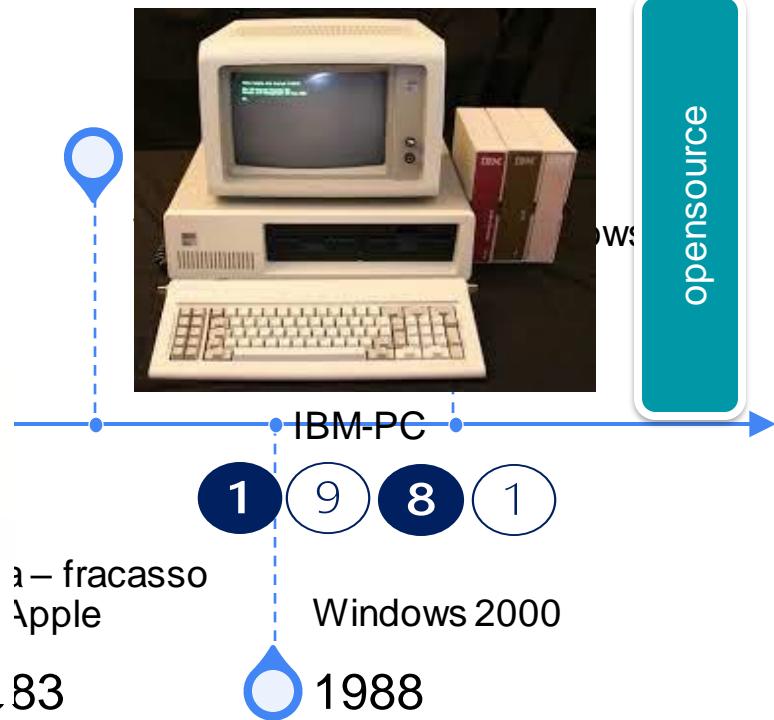
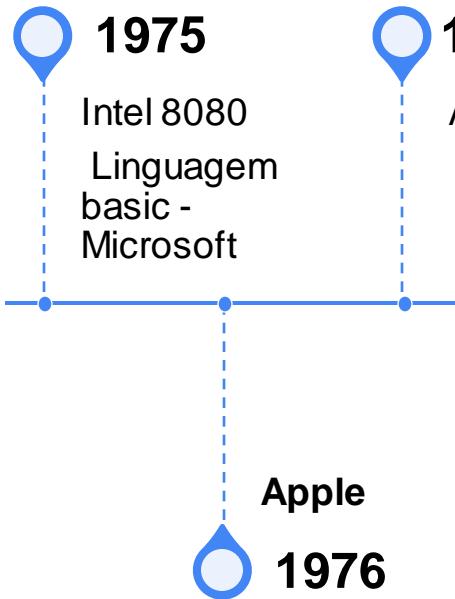
2001

Windows XP

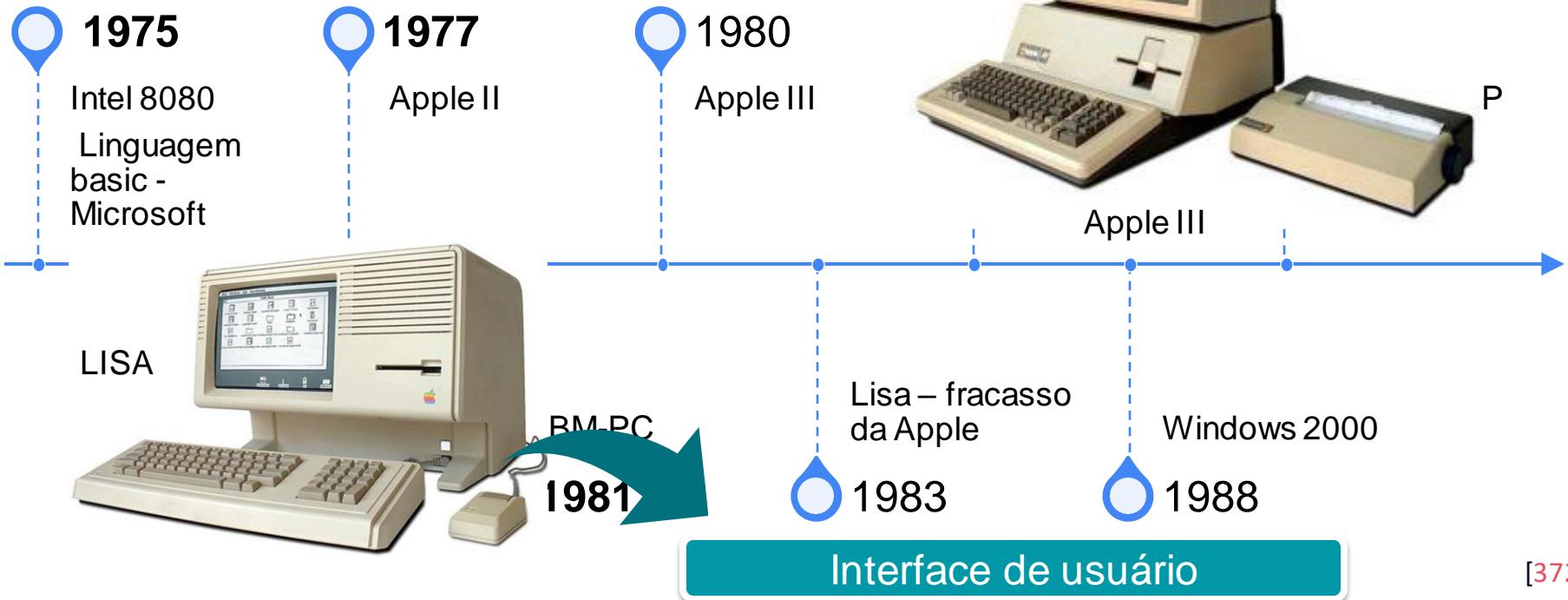
1988

Windows 2000

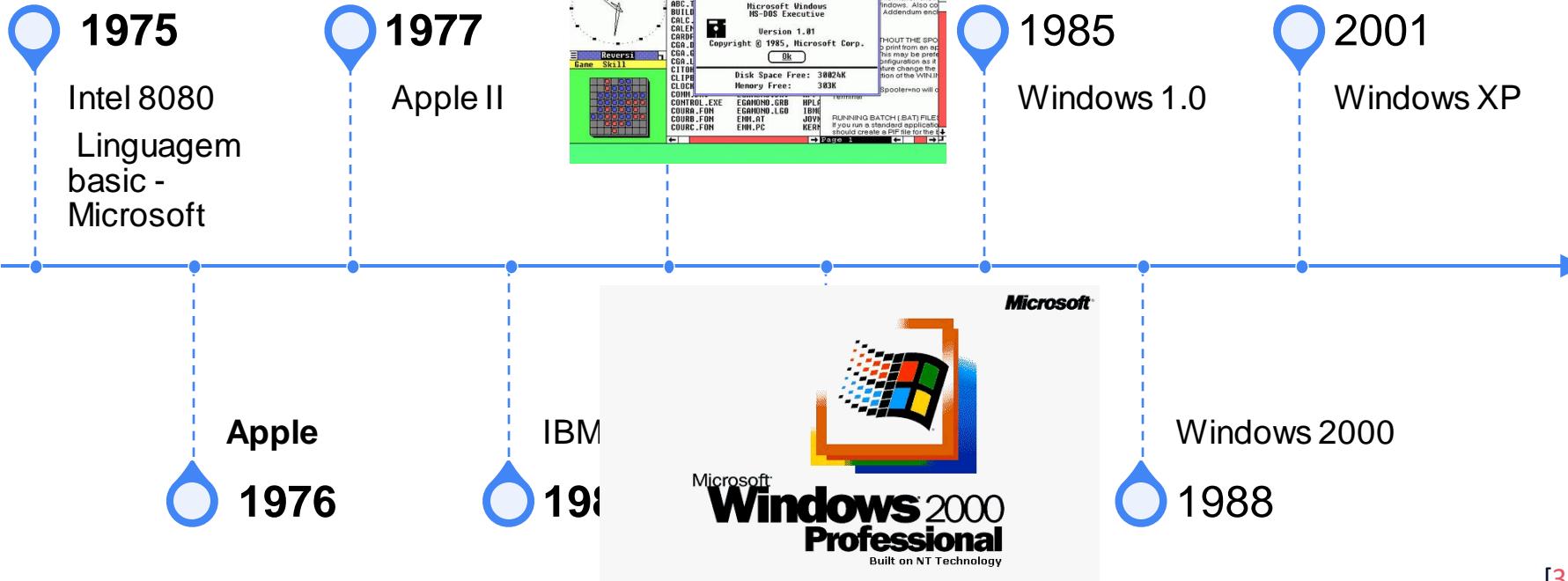
# Introdução



# Introdução



# Introdução



# Introdução

- 1949
- Primeira linguagem
- Linguagem de
  - Montagem
  - Máquina

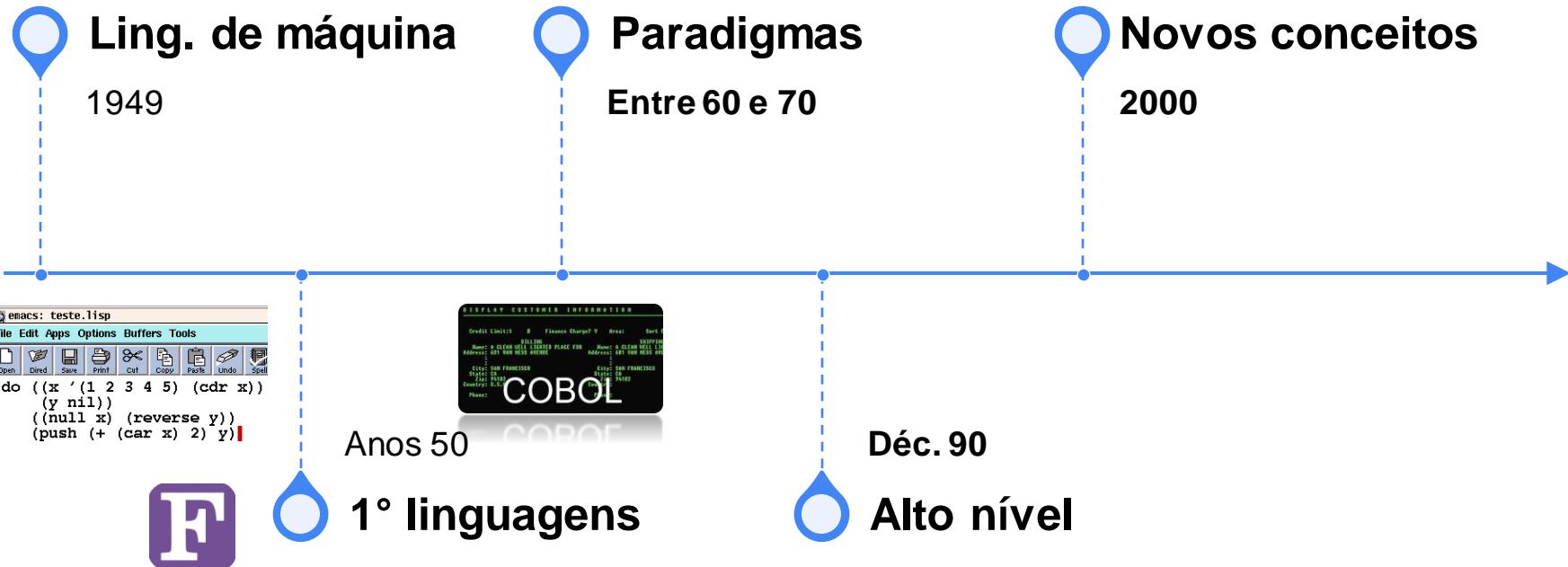
## Assembly

```
C014 24 FA      BCC     INCH      RECIEVE NOT READY
C016 B6 80 05    LDA A   ACIA+1   GET CHAR
C019 84 7F      AND A   #$7F    MASK PARITY
C01B 7E C0 79    JMP    OUTCH   ECHO & RTS
```

```
*****
* FUNCTION: INHEX - INPUT HEX DIGIT
* INPUT: none
* OUTPUT: Digit in acc A
* CALLS: INCH
* DESTROYS: acc A
* Returns to monitor if not HEX input
```

```
C01E 8D F0      INHEX   BSR     INCH      GET A CHAR
C020 81 30      CMP A   #'0    ZERO
C022 2B 11      BMI    HEXERR  NOT HEX
C024 81 39      CMP A   #'9    NINE
C026 2F 0A      BLE    HEXRTS  GOOD HEX
C028 81 41      CMP A   #'A    ...
C02A 2B 09      BMI    HEXERR  NOT HEX
```

```
BSR      INCH      GET A CHAR  
CMP A   # '0      ZERO  
BMI     HEXERR    NOT HEX  
CMP A   # '9      NINE  
BLE     HEXRTS    GOOD HEX  
CMP A   # 'A      ..  
BMI     HEXERR    NOT HEX
```



1 9 7 7



BSR      INCH  
 CMP A # '0  
 BMI    HEXERR  
 CMP A # '9  
 BLE    HEXRTS  
 CMP A # 'A  
 BMI    HEXERR  
 "



Ling

1949

```
emacs: teste.lisp
File Edit Apps Options Buffers Tools
Open Dired Save Print Cut Copy Paste Undo Spell
(do ((x '(1 2 3 4 5) (cdr x))
     (y nil)
     ((null x) (reverse y))
     (push (+ (car x) 2) y)
  )
```



Anos 50

1º linguagens



Paradigmas

Entre 60 e 70

Apple II

Novos conceitos

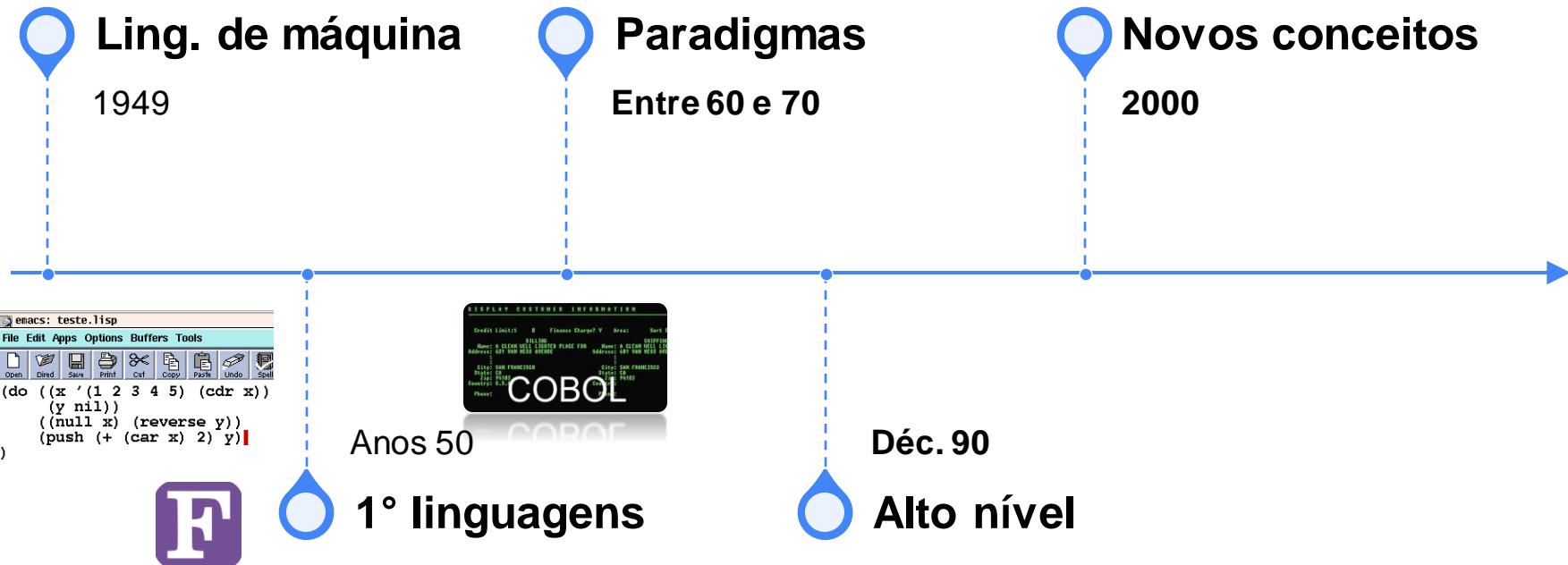
2000

Déc. 90

Alto nível



# Prolog

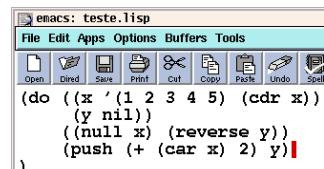




# Prolog

## Ling. de máquina

1949



## Paradigmas

Entre 60 e 70



Anos 50



1º linguagens

## Novos conceitos

2000

JavaScript



Déc. 90



Alto nível



```

BSR    INCH      GET A CHAR
CMP A #'0       ZERO
BMI    HEXERR   NOT HEX
CMP A #'9       NINE
BLE    HEXRTS   GOOD HEX
CMP A #'A       NOT HEX
BMI    HEXERR   NOT HEX
...
  
```



## Prolog



### Ling. de máquina

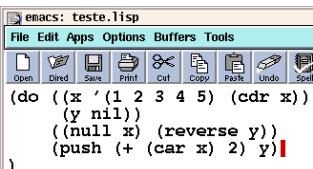
1949

### Paradigmas

Entre 60 e 70

### Novos conceitos

2000



Anos 50



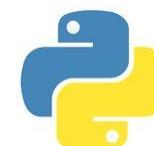
### 1º linguagens

### JavaScript



Déc. 90

### Alto nível



# Introdução

Problemas computacionais

Objeto de discussão que possui instruções passo a passo que são mais facilmente resolvíveis em ambiente computacional.

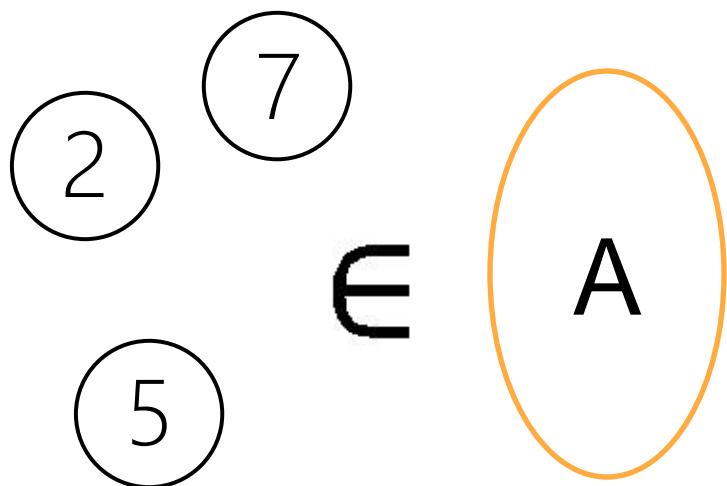
Problemas de decisão

Problemas de busca

Problemas de otimização

# Introdução

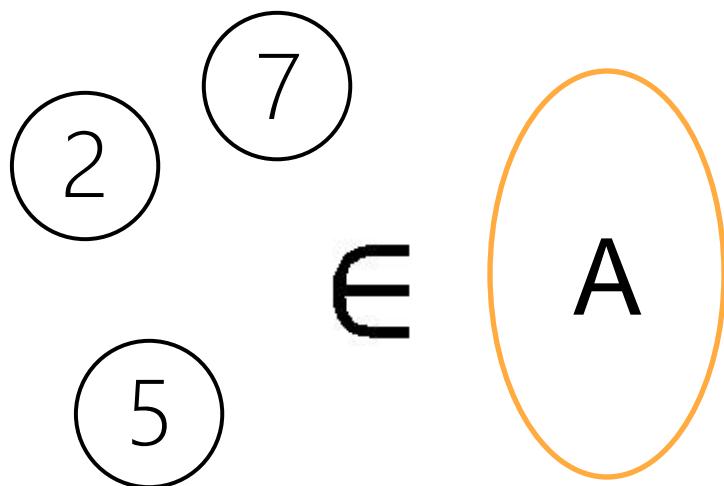
Problemas de decisão



- Caracter lógico - Sim ou Não
- Ideia
  - Pertencimento

# Introdução

Problemas de decisão

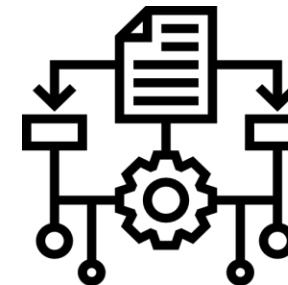
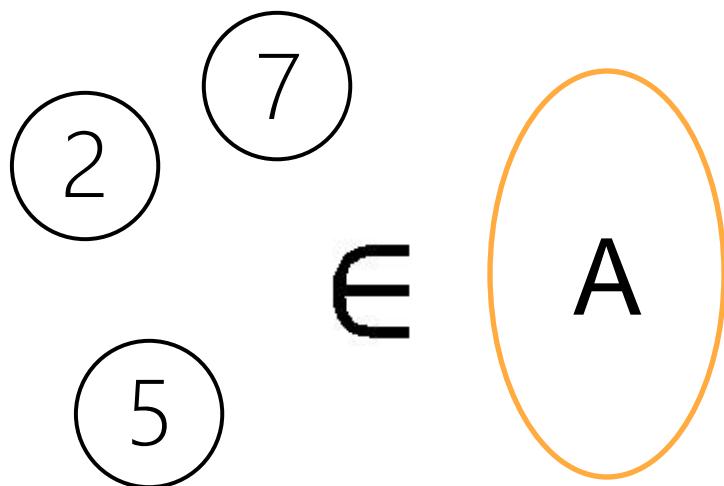


- Caracter lógico - Sim ou Não
- Ideia
- Pertencimento

**Ex: "Dado um número n inteiro positivo, determine se n é primo"**

# Introdução

Problemas de decisão



- Caracter lógico - Simulação
- Ideia
- Pertencimento

Problema decidível

Ex: "Dado um número  $n$  inteiro positivo, determine se  $n$  é primo"

# Introdução

## Problemas de busca



- Relacionamento binário
- Objetivo
  - Semelhante ao nome
  - $x$  está em A?
- Recorrente em teoria de grafos
  - **Ex: clique**

# Introdução

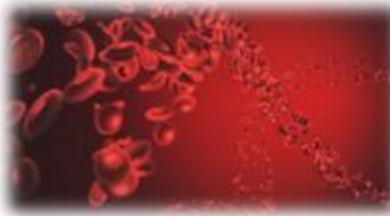
Problemas de otimização



Objetivo

- Maximizar, ou
- Minimizar uma função

Aplicações



# Introdução

O que é uma linguagem de programação?

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

# Introdução

O que é uma linguagem de programação?

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



Conjunto de palavras com regras

# Introdução

Código fonte

Tradução

Interpretação



## Etapa 2

# Como o computador entende o programa?

// Primeiros passos para começar a programar/  
Linguagens de programação

# Introdução

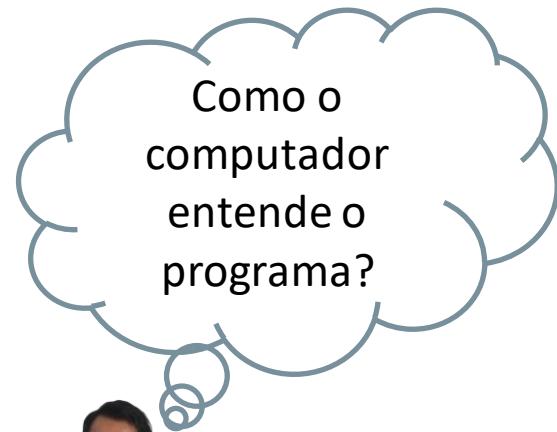
Código fonte

Tradução

Interpretação



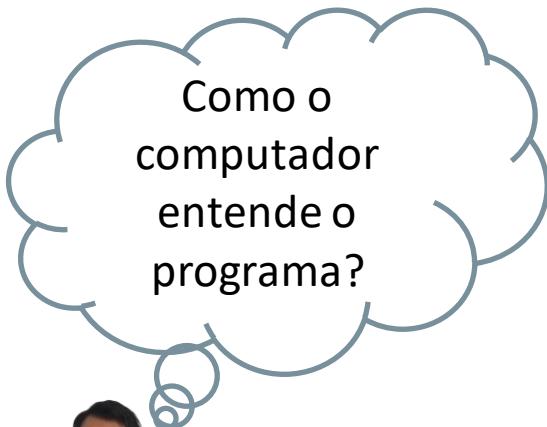
# Introdução



Como o  
computador  
entende o  
programa?

Um programa é um amontado de palavras  
senão for possível que o computador  
entenda

# Introdução



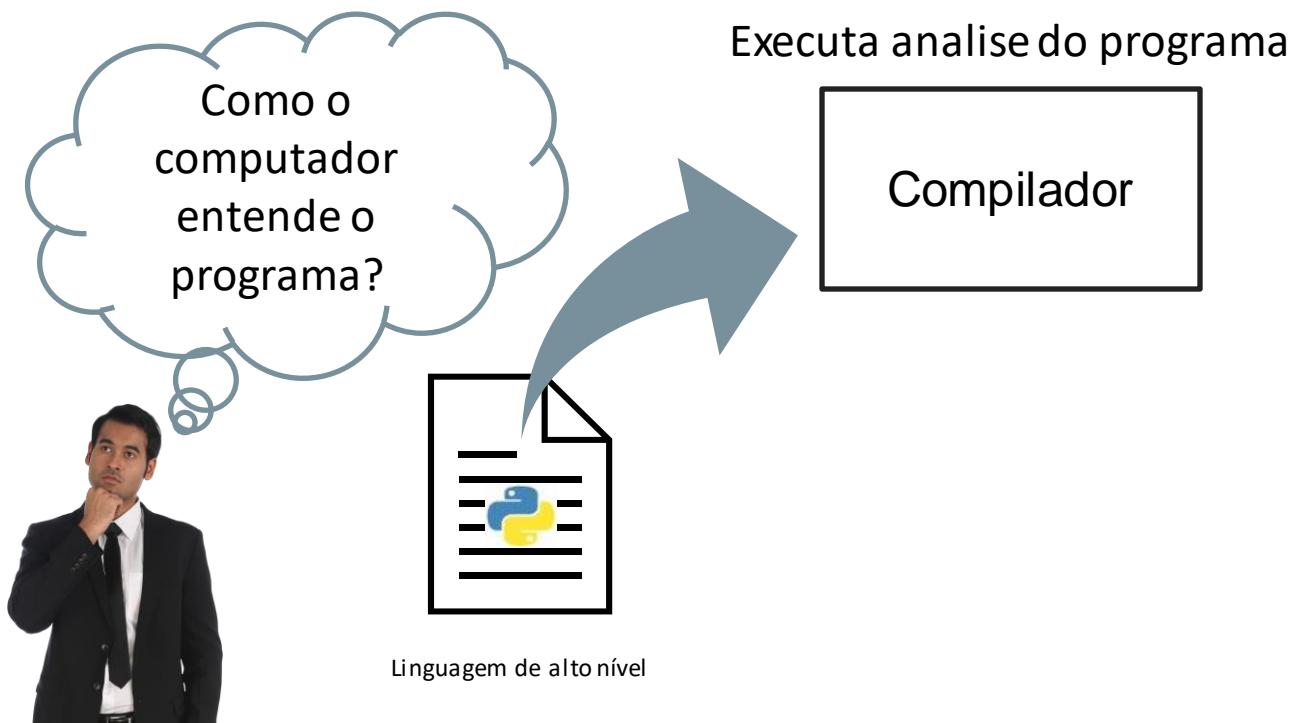
Como o computador entende o programa?

Um programa é um amontado de palavras senão for possível que o computador entenda



Processo de tradução

# Introdução



# Introdução

Como o computador entende o programa?



Linguagem de alto nível

Executa análise do programa



Compilador

```
C014 24 F0      BCC    INCH   RECEIVE NOT READY
C016 B6 80 05    LDA A  ACIA+1  GET CHAR
C019 84 7F      AND A #7F    MASK PARITY
C01B 7E C0 79    DRR A  INCH   ECHO & RTS

*****
* FUNCTION: INHEX - INPUT HEX DIGIT
* INPUT: none
*          Digit in acc A
* CALLS: INCH
* DESTRUCTS: acc A
*           Returns to monitor if not HEX input
C01E 8D P0      INHEX  BSR A  INCH   GET A CHAR
C020 81 30      BMI A  HEXERR  ZERO
C022 2B 11      BMI    HEXERR  NOT HEX
C024 81 39      CMP A #19    NINE
C026 81 0A      BLS A  INCH    GOOD HEX
C028 81 41      CMP A #7A
C02A 2B 09      BMI    HEXERR  NOT HEX
```

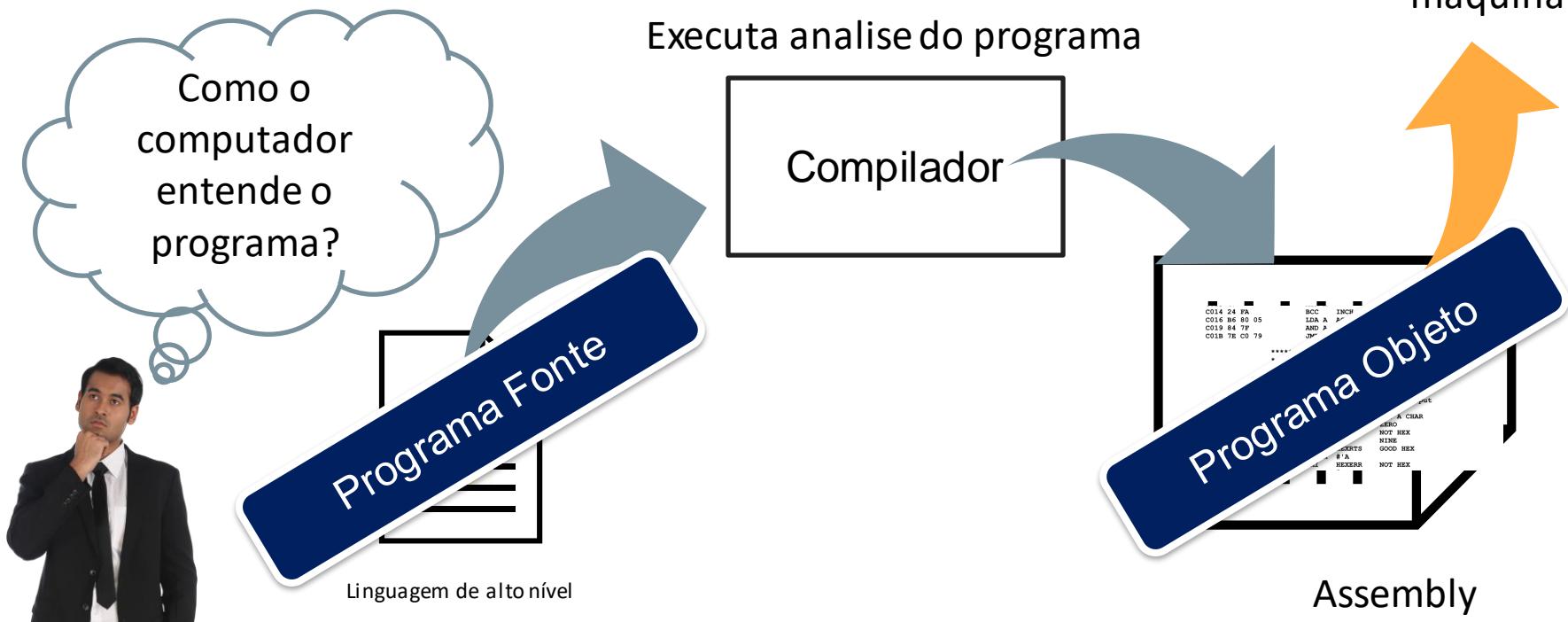
```
INHEX  BSR A  INCH   GET A CHAR
ZERO  BMI A  HEXERR  ZERO
NOT HEX BMI    HEXERR  NOT HEX
NINE   CMP A #19    NINE
GOOD HEX BLS A  INCH    GOOD HEX
NOT HEX BMI    HEXERR  NOT HEX
```

Linguagem de máquina



Assembly

# Introdução



# Introdução

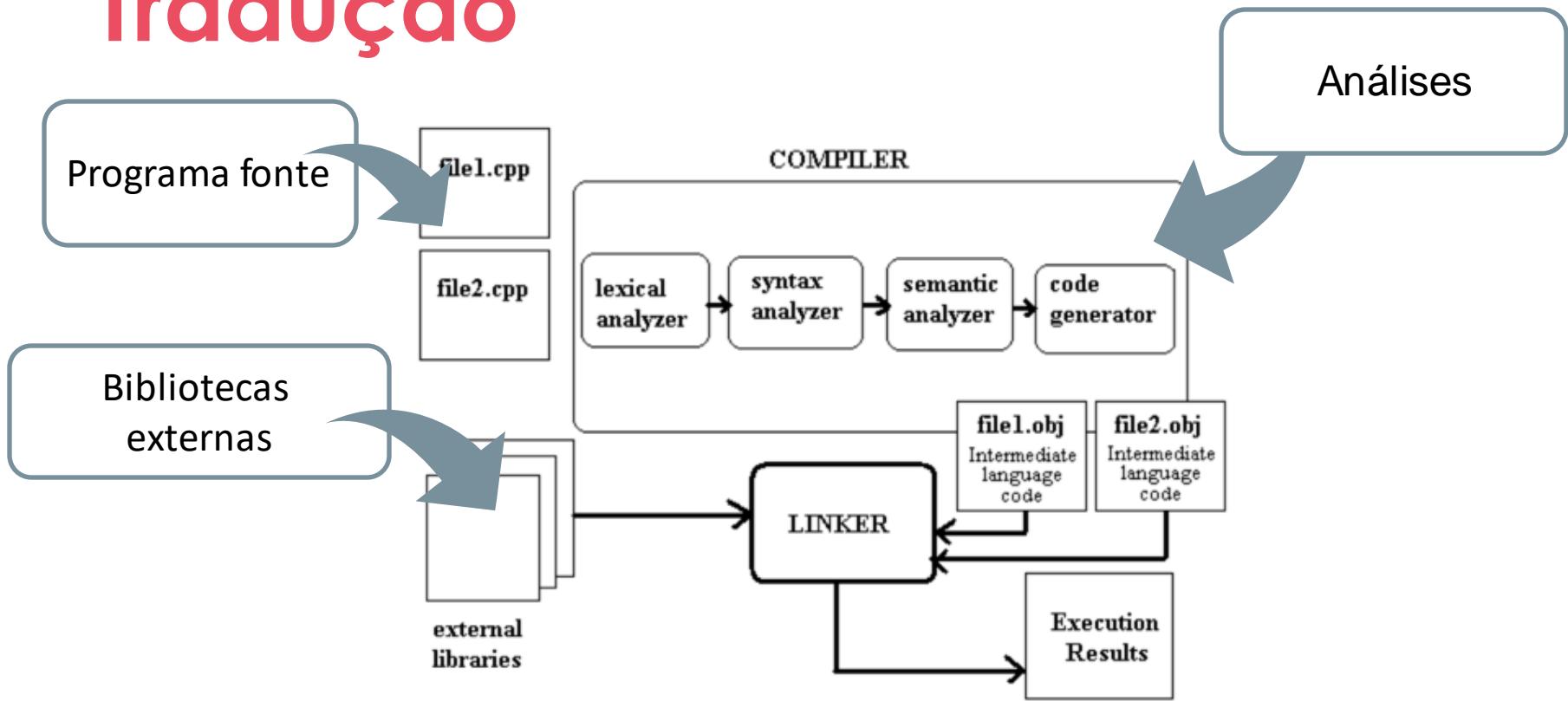
Tradução

- (1) Geração do programa objeto
- (2) Execução do programa objeto

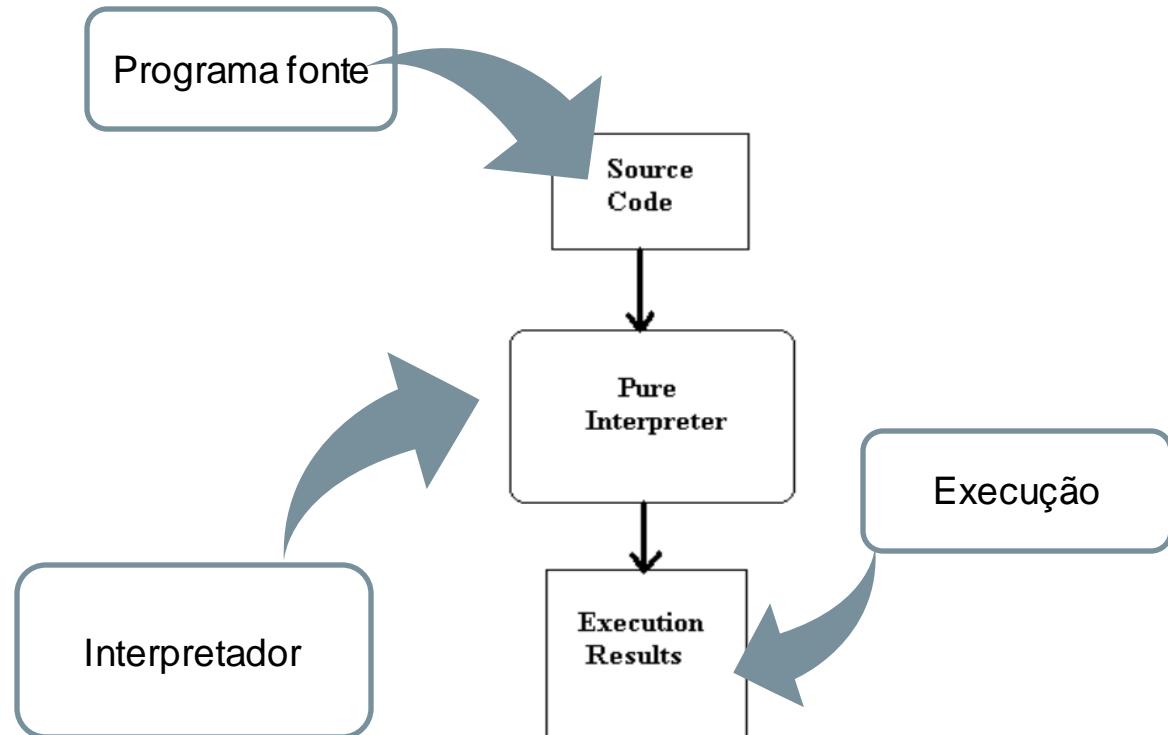
Interpretação

Programa fonte executado diretamente

# Tradução



# Tradução



# Introdução

Tradução

- 
- (1) Geração do programa objeto
  - (2) Execução do programa objeto

Interpretação

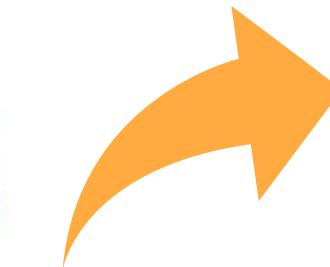
Programa fonte executado diretamente

Maior flexibilidade

# Introdução



Tradução



Execução mais rápida

- (1) Geração do programa objeto
- (2) Execução do programa objeto

JavaScript



Interpretação



Programas menores

Programa fonte executado diretamente



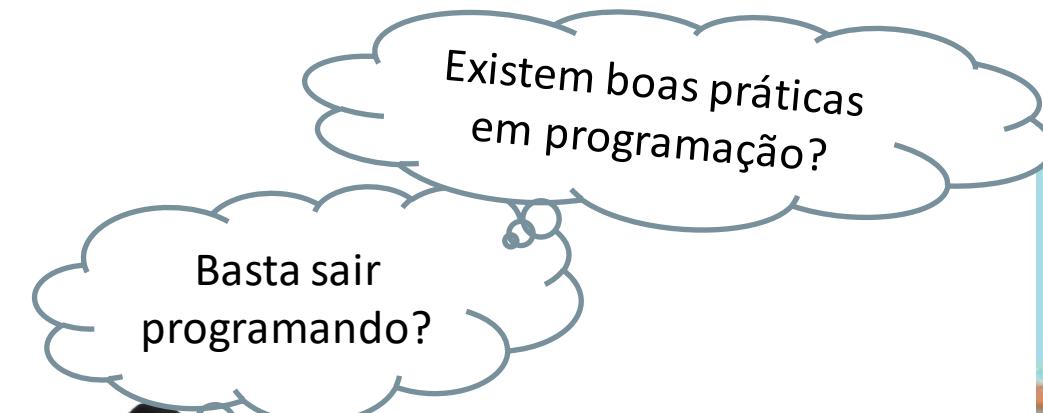
Maior flexibilidade

## Etapa 3

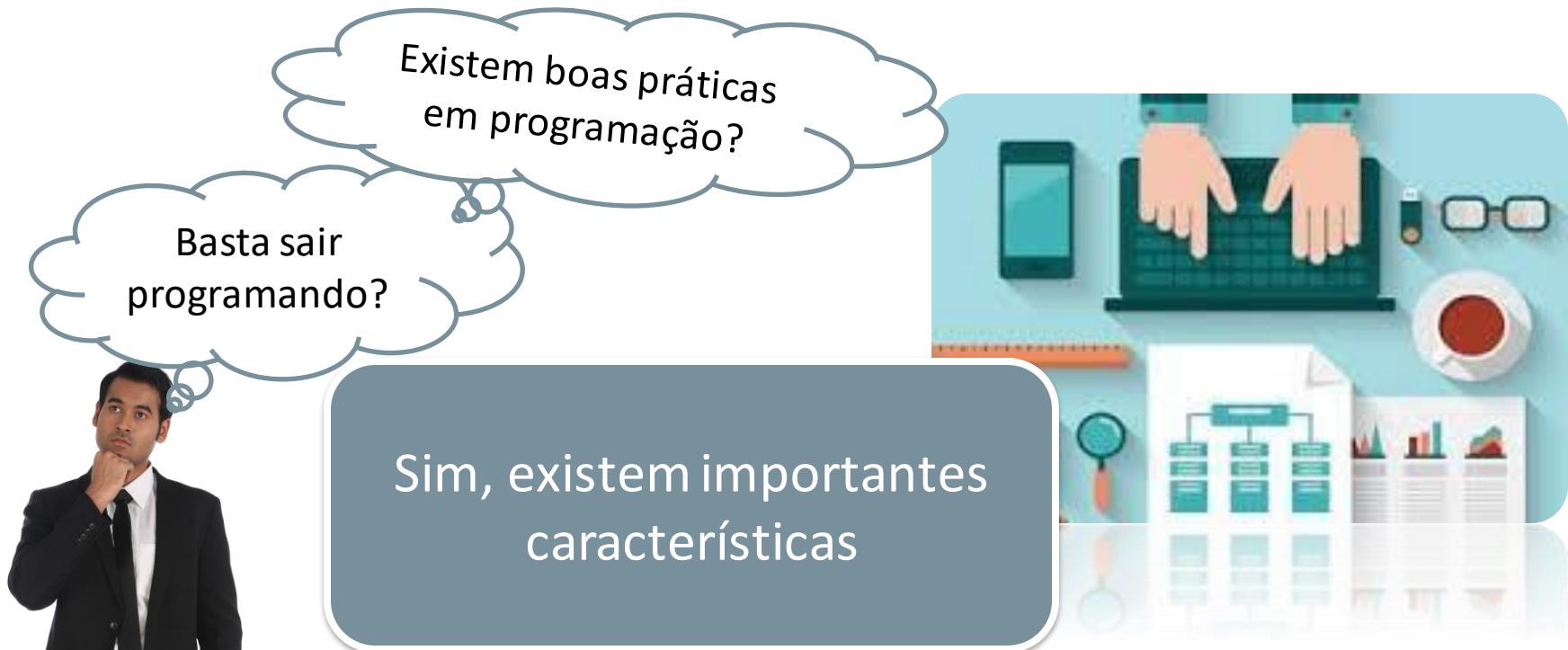
# Características de um programa

// Primeiros passos para começar a programar/  
Linguagens de programação

# Características



# Características



A man in a suit is shown from the waist up, looking thoughtful with his hand to his chin. He has a thought bubble above his head containing two questions: "Existem boas práticas em programação?" and "Basta sair programando?". A large blue rounded rectangle contains the answer: "Sim, existem importantes características". To the right of the man is a light blue desk with a laptop, a smartphone, a pair of glasses, a notepad, a ruler, a magnifying glass over a code snippet, and a document with a data flow diagram.

Existem boas práticas em programação?

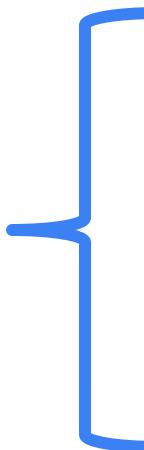
Basta sair programando?

Sim, existem importantes características

# Características

Desenvolvimento de  
programas

Diretrizes



Legibilidade

Redigibilidade

Confiabilidade

Custo

# Características

Legibilidade

- Facilidade de leitura
- Compreensão
- Ortogonalidade
- Definição adequada das estruturas



"Estado que é legível"

Coerência nas instruções

# Características

## Redigibilidade

- Pode conflitar com a legibilidade
- Ortogonalidade
- Simplicidade da escrita
- Suporte à abstração
- Reuso do código
- Expressividade



Facilidade de escrita de  
código

Coerência nas instruções

# Características

## Redigibilidade

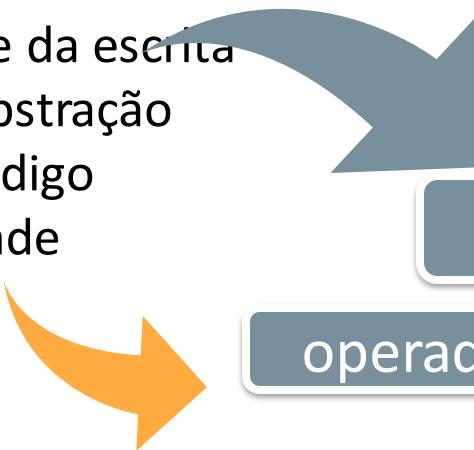
- Pode conflitar com a legibilidade
- Ortogonalidade
- Simplicidade da escrita
- Suporte à abstração
- Reuso do código
- Expressividade

Facilidade de escrita de código

Coerência nas instruções

operador ++

uso do for



# Características

Confiabilidade

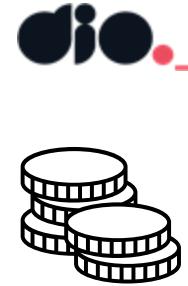
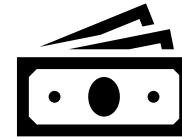
Possui:

- verificação de tipos
- Trata exceções
- Uso de ponteiros
- Compatibilidade entre compiladores

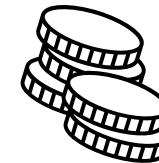


Faz o que foi programado para fazer

# Características



Custo



Análise de impacto

Possui:

- Treinamento
- Codificação
- Compilação
- Execução
- Infra-estrutura



# Características

Outras características

Atualizações

Uso para IA

Disponibilidade de ferramentas

Comunidade ativa

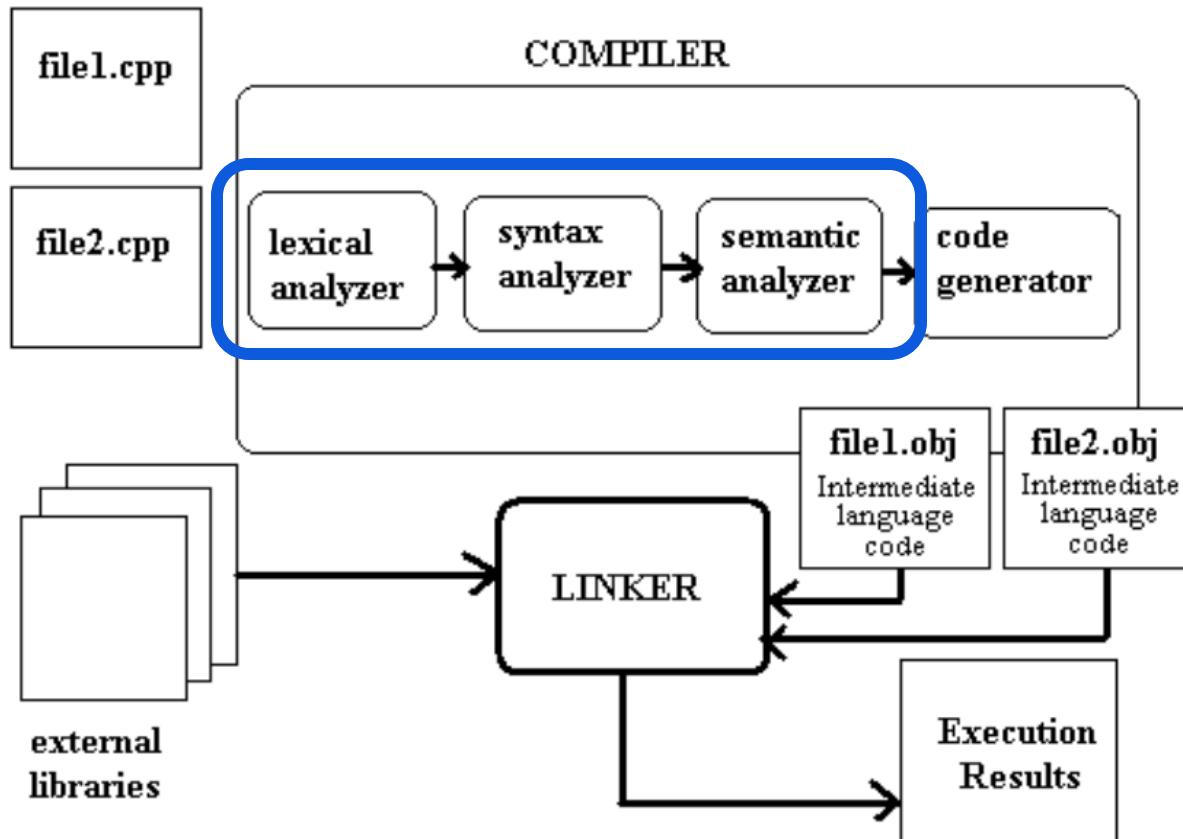
Adoção pelo mercado

## Etapa 4

# Análises de código

// Primeiros passos para começar a programar/  
Linguagens de programação

# Análise



# Análise léxica

Particionar

Classificar

Eliminar

# Análise léxica



Elementos  
denominados tokens

# Análise léxica



# Análise léxica

Particionar

Classificar

Eliminar

Elementos  
denominados tokens

Elementos:

identificadores, palavras reservadas,  
números, strings

Elementos:

Caracteres de espaços em  
branco, comentários, ...

# Análise sintática

Forma

Sintaxe:

"componente do sistema linguístico que interligam os constituintes da sentença, atribuindo-lhe uma estrutura"

Corretude do programa

# Análise sintática

Forma

Sintaxe:

"componente do sistema linguístico que interligam os constituintes da sentença, atribuindo-lhe uma estrutura"

Padrão - gramática

- Depende da linguagem de programação utilizada

# Análise semântica

Significado

Semântica:

"É o estudo do significado. Incide sobre a relação entre significantes, como: palavras, frases, sinais e símbolos"

Lógica do programa

# Análise semântica

Significado

Semântica:

"É o estudo do significado. Incide sobre a relação entre significantes, como: palavras, frases, sinais e símbolos"

Erro de semântica:

- não faz o que é esperado

```
if (x = 0)  
printf("O valor e' nulo\n");
```

# Análise semântica

Significado

Semântica:

"É o estudo do significado. Incide sobre a relação entre significantes, como: palavras, frases, sinais e símbolos"

Erro de semântica

Erro de semântica:

- não faz o que é esperado

```
if (x == 0)  
printf("O valor e' nulo\n");
```

```
if (x = 0)  
printf("O valor e' nulo\n");
```

## Etapa 5

# Paradigmas de programação

// Primeiros passos para começar a programar/  
Linguagens de programação

# Paradigmas



O que é um  
paradigma?

# 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."

## Classificação

Orientação à objeto

Procedural

Funcional

Estruturado

Computação  
distribuída

Lógico

Chamadas sucessivas e procedimentos separados

## Classificação

Orientação à objeto

Procedural

Funcional

Estruturado

Computação distribuída

Lógico

## Classificação

Orientação à objeto

Procedural

Funcional

Estruturado

Computação  
distribuída

Lógico

InSTRUÇÕES SÃO BASEADAS  
EM FUNÇÕES



## Classificação

Estrutura de blocos  
aninhados

Orientado à obj...

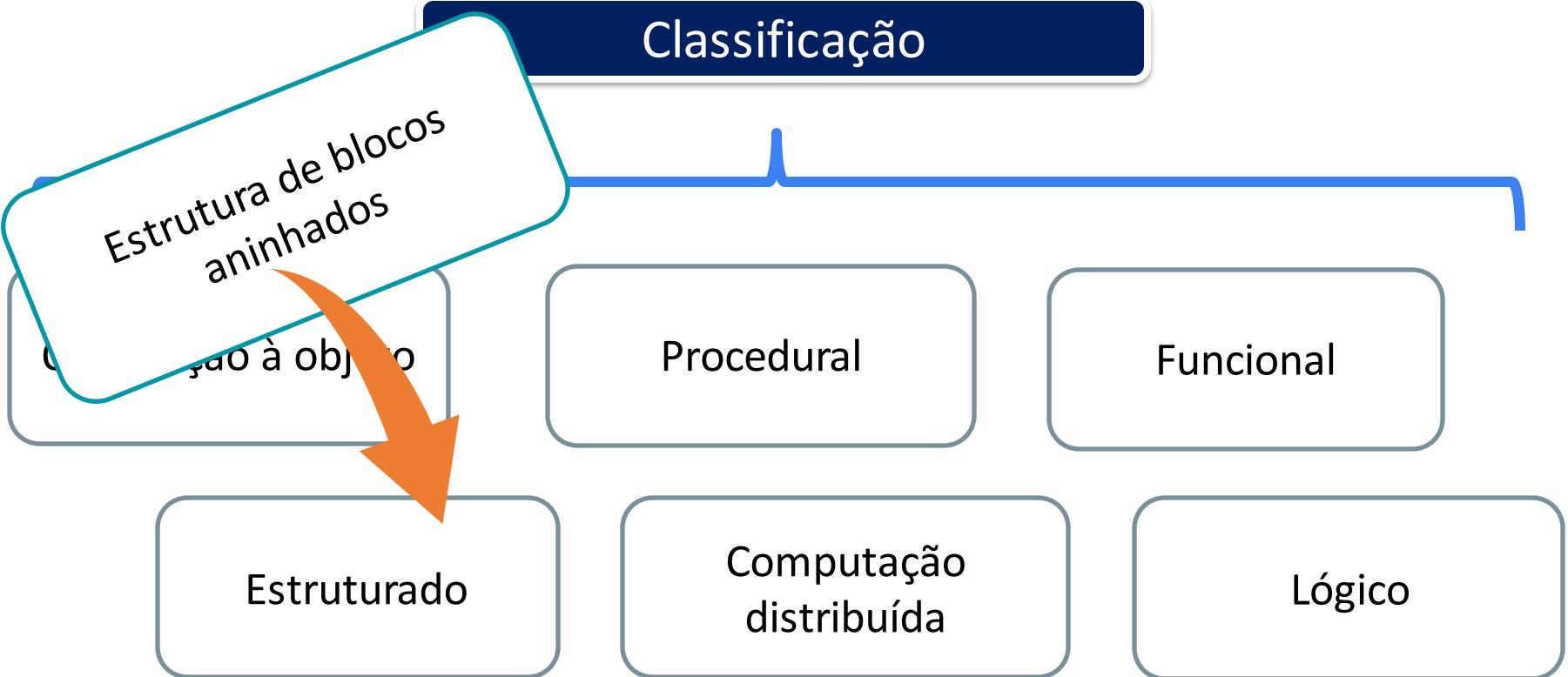
Procedural

Funcional

Estruturado

Computação  
distribuída

Lógico



## Classificação

Funções executadas de forma independente  
Orientação a objeto

## Classificação

Orientação à objeto

Procedural

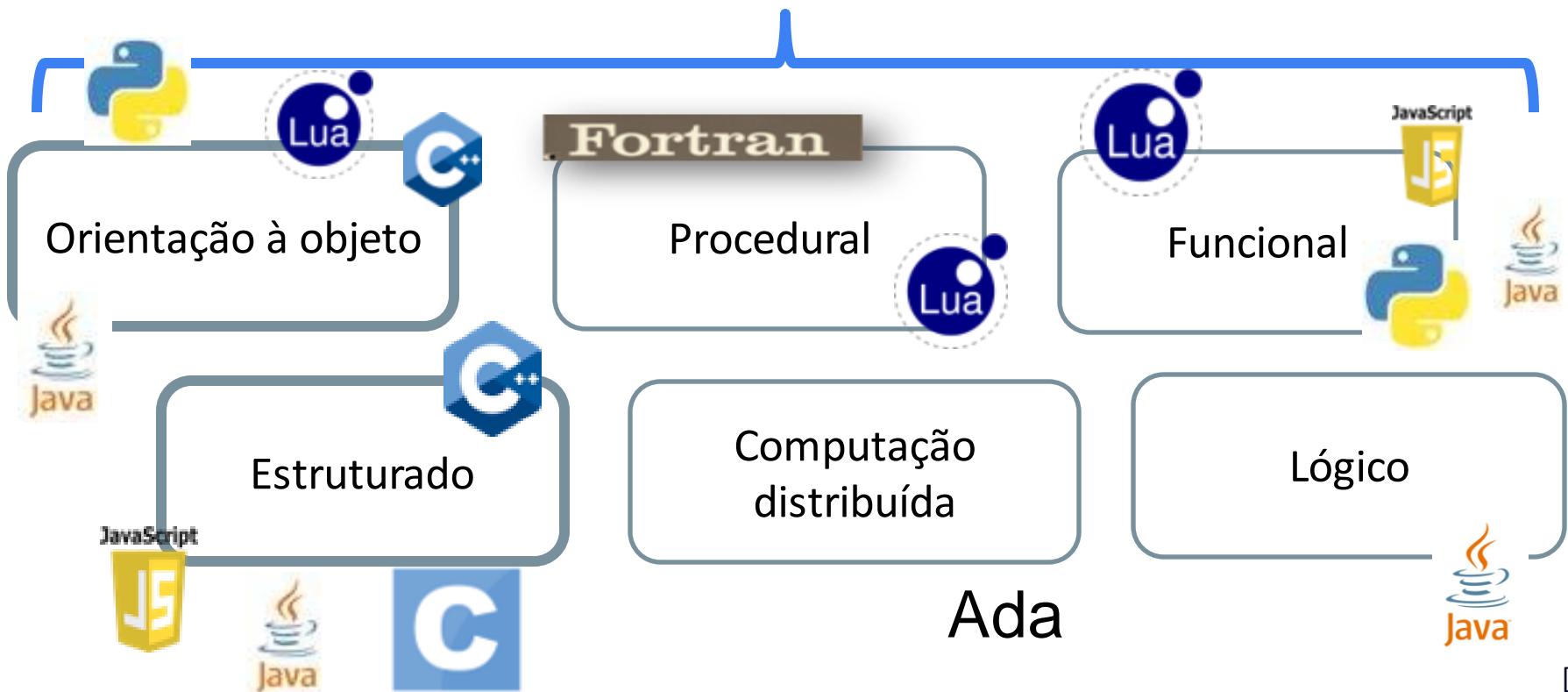
Funcional

Estruturado

Computação  
distribuída

Lógico

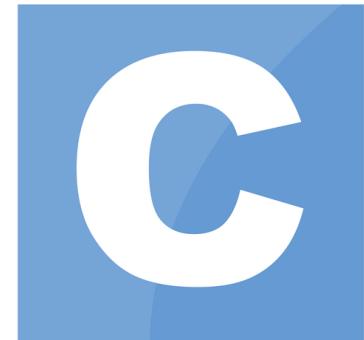
# Classificação



# Paradigma Estruturado

Conceitos:

- Sequência
- Decisão
- Iteração



Ênfase em sequência

# Paradigma Estruturado

Conceitos:

- Sequência
- Decisão
- Iteração



Teste lógico



Ênfase em sequência

# Paradigma Estruturado

Conceitos:

- Sequência
- Decisão
- Iteração

Dominou o mercado até POO



Funções, laços, condições

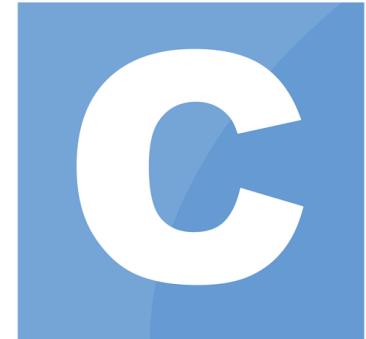


Ênfase em sequência

# Paradigma Estruturado

Utilização:

- Problemas simples e diretos
- Aprender programação



POO ainda não compreendida por muitos

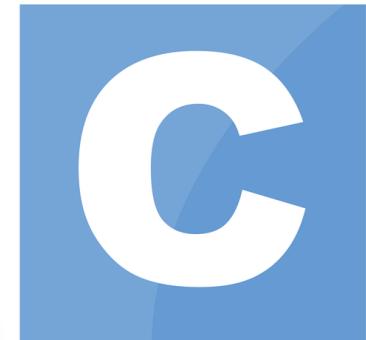
Ênfase em sequência

# Paradigma Estruturado

Conceitos:

- Sequência
- Decisão
- Iteração

```
function factorial(x){  
    if(x > 1){  
        return x*fatorial(x-1);  
    }  
}
```



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?



# Orientação à Objetos

O que é um objeto?



# Orientação à Objeto

O que é um objeto?

"Um objeto é descrito por características específicas, comportamentos e estado"

O que tenho

Sou capaz de fazer

Como faço

# Orientação à Objeto

O que tenho

Modelo  
Cor  
Carga  
Corpo  
Tampa  
Ponta

Sou capaz de fazer

escrever  
desenhar  
rabiscar  
pintar  
destampar

Como faço

Tampada  
Destampada  
Em uso



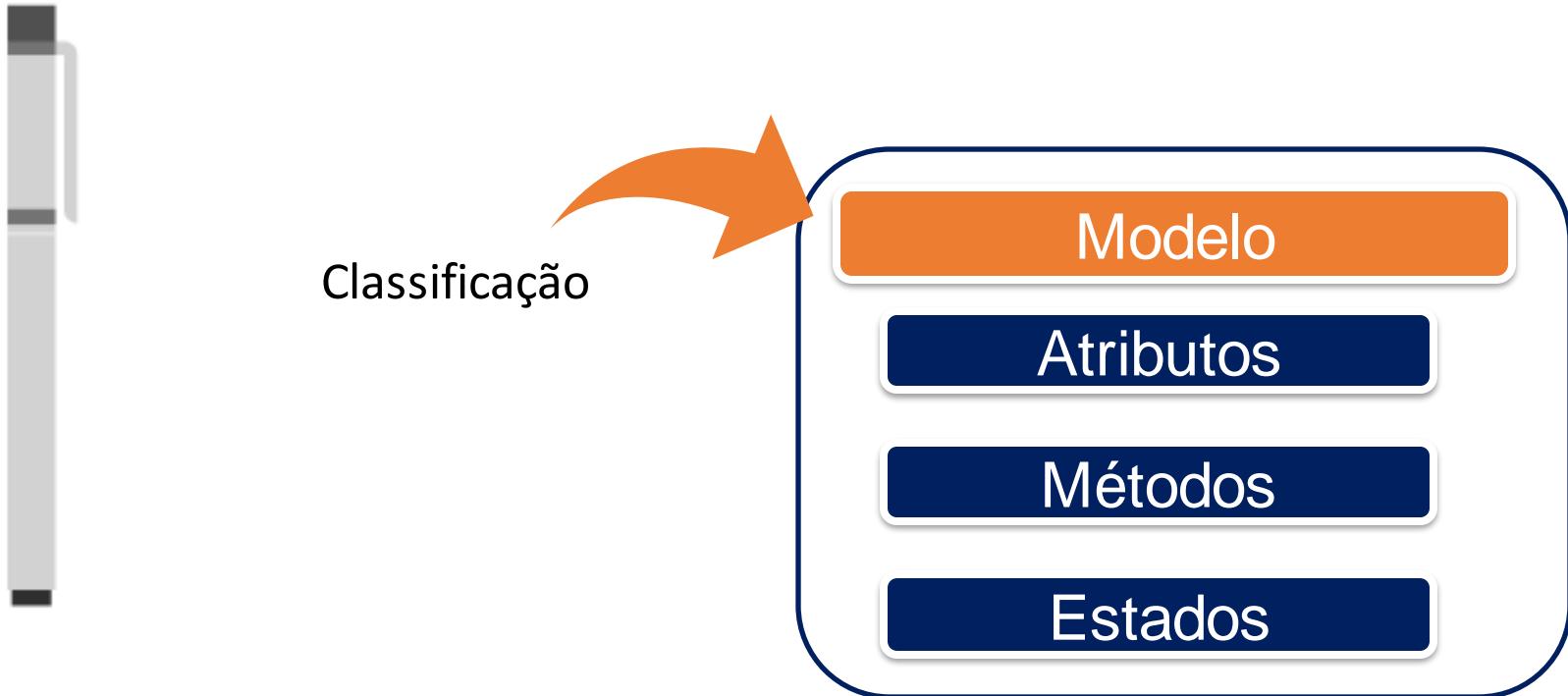
# Orientação à Objeto



- O que tenho
- Sou capaz de fazer
- Como faço



# Orientação à Objeto



# Orientação à Objeto

O que é um objeto?

Ponto de vista da programação

Classe

Alocação em memória

Operações associadas

POO

Estruturada

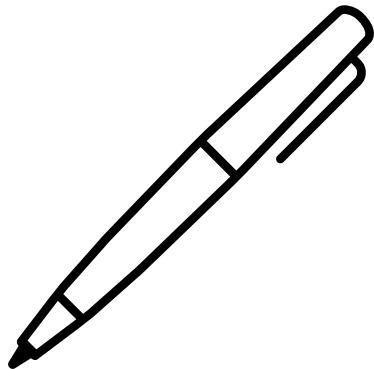
Alocação em  
memória

Operações  
associadas

Variável



# Orientação à Objeto



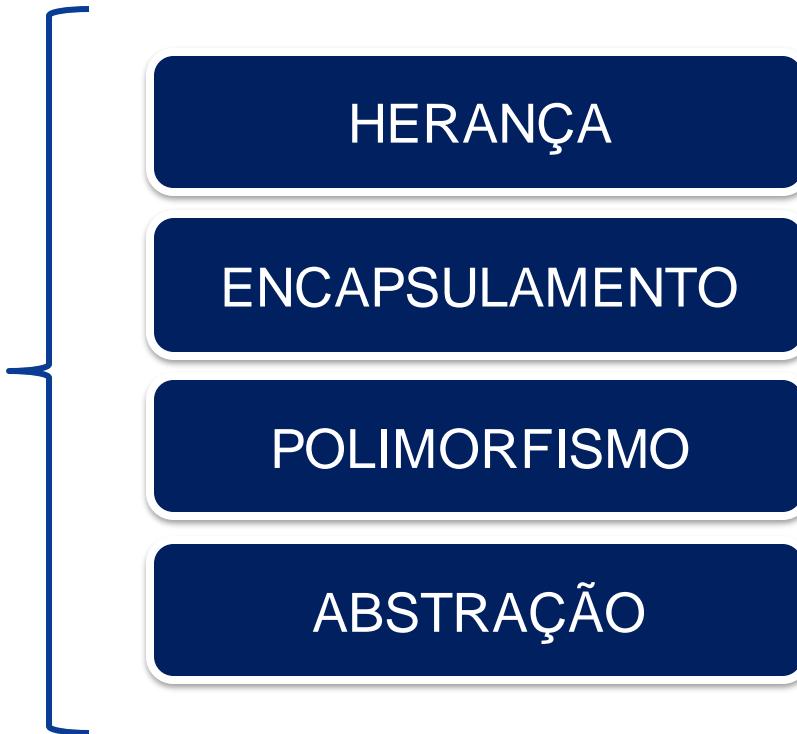
```
classe caneta():
    cor: inteiro
    carga: caracter
    tampada: lógico

    metodo escrever()
        Se estado tampada
            Escreva ("Destampar")
        Senão
            Escreva ("algo")
        Fimse

    metodo tampar()
        Tampada = verdadeiro
fimclasse
```

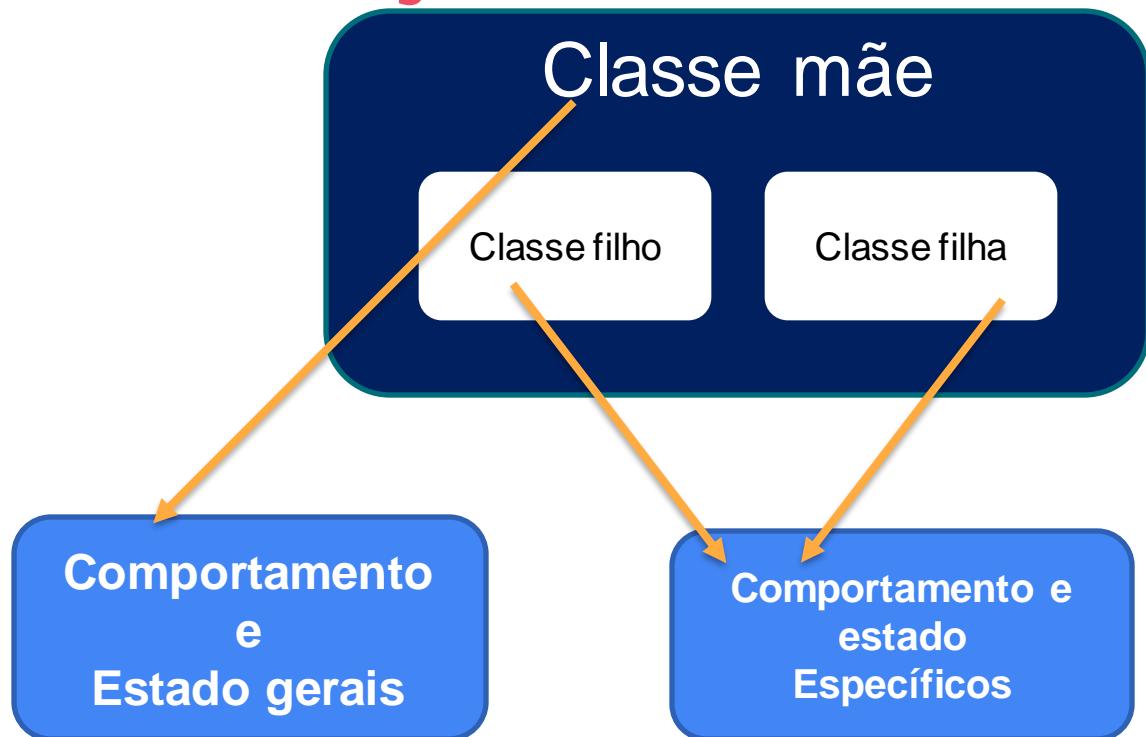
# Orientação à Objeto

Pilares de  
Orientação a Objeto

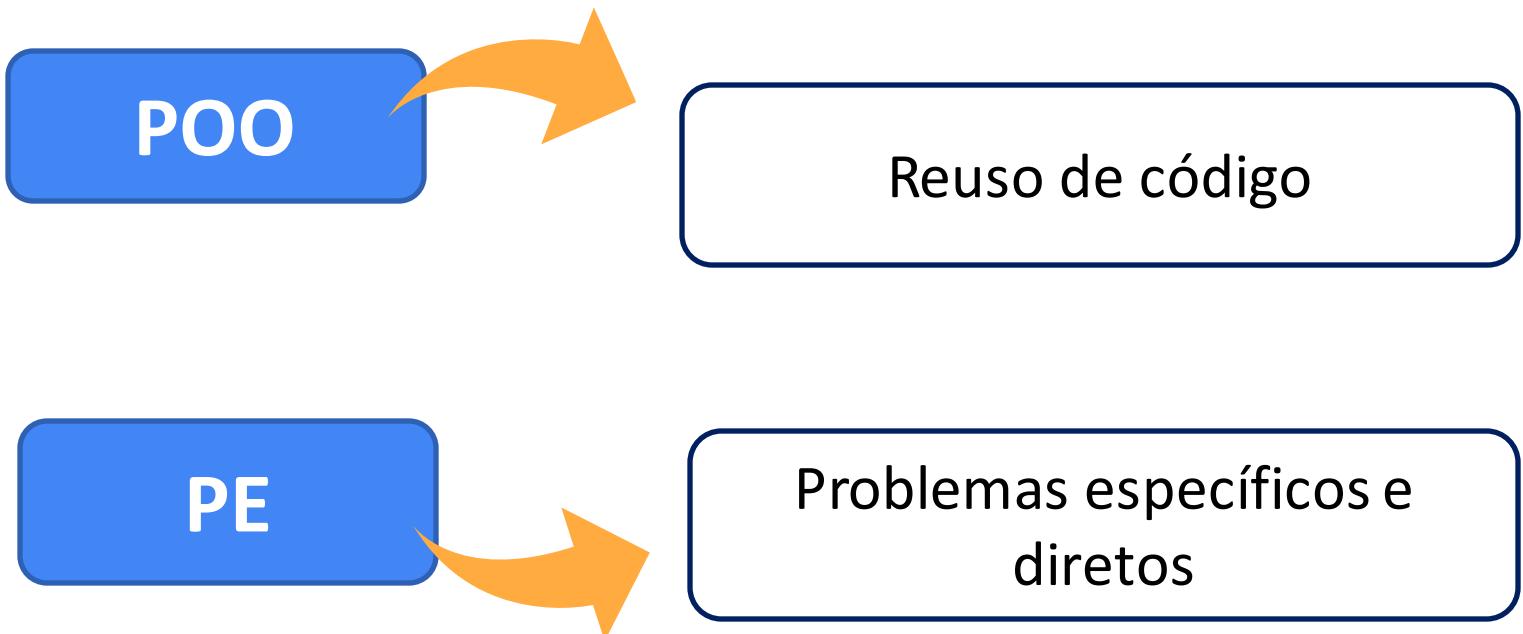


# Orientação à Objeto

- Classes
- Características
  - Atributos
  - Métodos



# Orientação à Objeto



# Referências Bibliográficas

// Primeiros passos para começar a programar

# Referências

## Computational Thinking

<https://www.cs.cmu.edu/~CompThink/resources/TheLinkWing.pdf>

<https://ubiquity.acm.org/article.cfm?id=1922682>

<https://www.youtube.com/watch?v=YVEUOHw3Qb8>

<https://simplificandoredes.com/historia-da-computacao/>

<https://simplificandoredes.com/pensamento-computacional/>

# Referências

## Padrões

<https://www.cin.ufpe.br/~if114/Monografias/Reconhecimento/>

<https://web.fe.up.pt/~jmsa/recpad/index.htm>

[https://www.teses.usp.br/teses/disponiveis/18/18133/tde-10072006-002119/publico/Capitulo\\_2\\_mestrado.pdf](https://www.teses.usp.br/teses/disponiveis/18/18133/tde-10072006-002119/publico/Capitulo_2_mestrado.pdf)

<http://www.vision.ime.usp.br/~teo/publications/dissertacao/node9.html>

<https://periodicos.furg.br/vetor/article/view/3363/3811>

# Referências

Programação e algoritmos

<http://www.inf.ufes.br/~tavares/labcomp2000/aulas.htm>

<https://www.dca.fee.unicamp.br/cursos/EA876/apostila/HTML/>

<http://www.inf.ufes.br/~tavares/labcomp2000/aula1.htm>

# Referências

## Algoritmos

Livro: Fundamentos da programação de computadores – Pearson

[https://www.youtube.com/watch?v=HtSuA80QTy&list=PLUl4u3cNGP61Oq3tWYp6V\\_F-5jb5L2iHb](https://www.youtube.com/watch?v=HtSuA80QTy&list=PLUl4u3cNGP61Oq3tWYp6V_F-5jb5L2iHb)

Algoritmos - MIT

# Referências

- José Augusto Manzano, Algoritmos - lógica para desenvolvimento de programação de computadores, Ed. Érica, 17a ed. (ou mais recente) - livro de referência
- Ana Fernandes Ascencio, Fundamentos da programação de computadores - algoritmos, Pascal, C/C e Java, Ed. Pearson, 2a ed.
- Andrew S. Tanenbaum, “Livro Organização Estruturada de Computadores”, Ed. Pearson, 5a edição -- Capítulo 1

# Referências

- The Formal Semantics of Programming Languages, Glynn Winskel Deitel e Deitel, “C++ How to Program”, 5th edition, Editora Prentice Hall, 2005 -- Capítulo 1
- Waldemar Celes, Renato Cerqueira e José Lucas Rangel, "Introdução a Estrutura de Dados com Técnica de Programação em C", Editora Campus-Elsevier, 2004 -- Capítulo 1

# Dúvidas?

- > Fórum/Artigos
- > Comunidade Online (Discord)

