

CONCEITOS BÁSICOS

Agenda

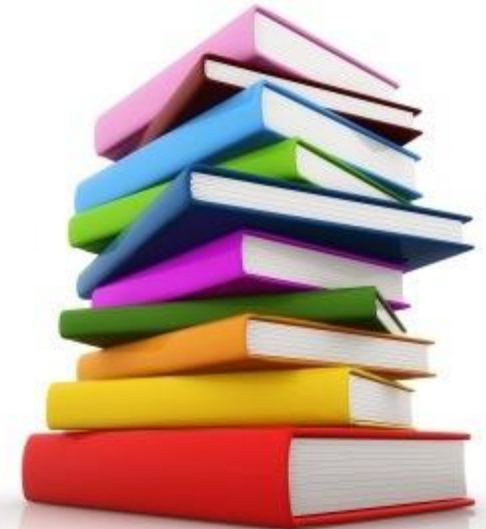
O que iremos aprender:

1. Conceitos Básicos sobre Hardware/Software
2. Construção de um Programa
3. Definição de Algoritmo
4. Formas de Representação dos Algoritmos
5. Estruturas de Controle
6. Definição de Programa
7. Compilador e Interpretador

Conceitos Básicos

✓ Hardware;

✓ Software;



Conceitos Básicos

Hardware

“tudo que é **físico** (concreto), ou seja, pode ser tocado”

Hardware



Conceitos Básicos

Software

“tudo que é **lógico** (abstrato), ou seja, **não** pode ser tocado”

Software



Software

Como desenvolver?!

Resolvendo problemas

1. Trocar o pneu de um carro
2. Trocar uma lâmpada
3. Determinar o melhor caminho de casa até o trabalho
4. ?????

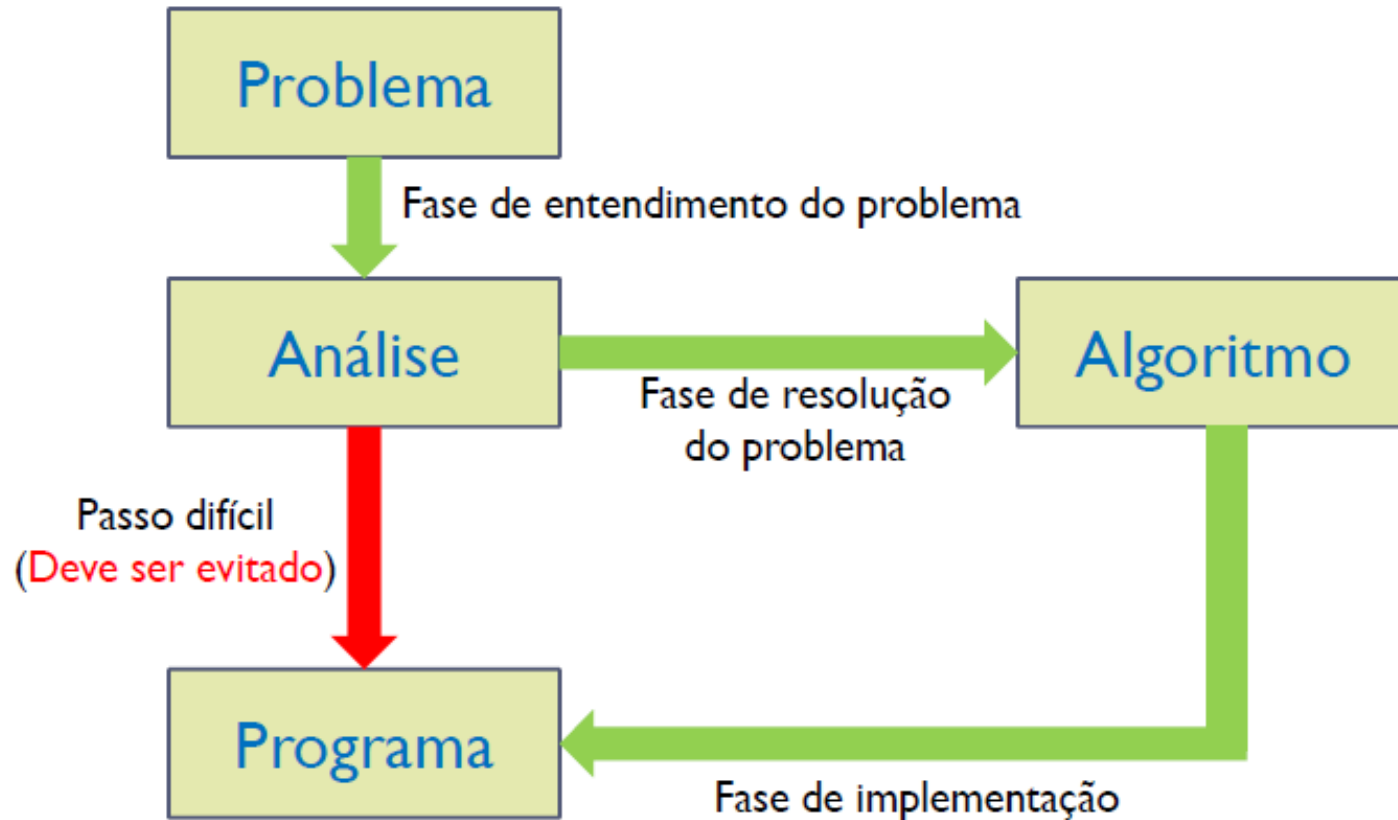
Qual seria uma boa solução?!

Construindo um programa

Problema

Programa

Etapas para Construção de um programa



Adaptado de (TREMBLAY, 1983)

Processo de Geração de um Programa (I)

1. **Problema:** o que se deseja resolver/satisfazer.
2. **Análise do Problema:** conjunto de atividades relacionadas ao entendimento do problema.
 - **Ler** atentamente o enunciado do problema **até entendê-lo**
 - Identificar os **dados de entrada**
 - Identificar as **saídas** (resultados esperados)
 - Identificar **o que** o programa deve fazer (seu objetivo)
 - Identificar se existem **valores ou dados intermediários** necessários para transformar entradas em saídas

Processo de Geração de um Programa (II)

1. Analisado o Problema, devemos pensar em como **estruturar uma solução viável** (dentre as muitas disponíveis)!
2. Para resolver um problema no computador é necessário que seja, primeiramente, encontrada uma maneira de descrever este problema de uma **forma clara e precisa!**
3. É preciso definir o **caminho** a ser tomado...



Como Desenvolver Software?



Algoritmo
(plano de ação)



Programa
(linguagem de programação)

LINHA DO TEMPO

Algoritmo

BUGIO, O PAI 2

POR WILLIAN RAPHAEL SILVA



Tirinha do site [Humor com Ciência](http://Humor.com/Ciência)

Algoritmo - Definição

É uma sequência **finita** de **passos lógicos** necessários para **realizar** uma determinada tarefa

**É POSSÍVEL ENCONTRAR DIFERENTES ALGORITMOS PARA
A RESOLUÇÃO DE UM PROBLEMA!**

Exemplos

Características Fundamentais

1. Ter início e fim.
2. Ser escrito em termos de ações ou comandos bem definidos (não ambíguo).
3. Ter uma sequência lógica.
4. Ter capacidade de receber dados de entrada.
5. Poder gerar informações de saída.
6. Ser efetivo (deve sempre resolver o que tem para solucionar).

Os algoritmos estão presentes no dia-a-dia!

Preparo de Bolo (dado os ingredientes disponíveis), segue o modo de preparo:

1. Bata a margarina, as gemas e o açúcar até ficar cremoso
2. Junte o leite e a farinha e continue batendo
3. Acrescente o fermento e as claras em neve
4. Unte a forma com manteiga e leve ao forno para assar

É possível fazer o bolo? Existem elementos de imprecisão?

Os algoritmos estão presentes no dia-a-dia!

Preparo de Bolo (dado os ingredientes disponíveis), segue o modo de preparo:

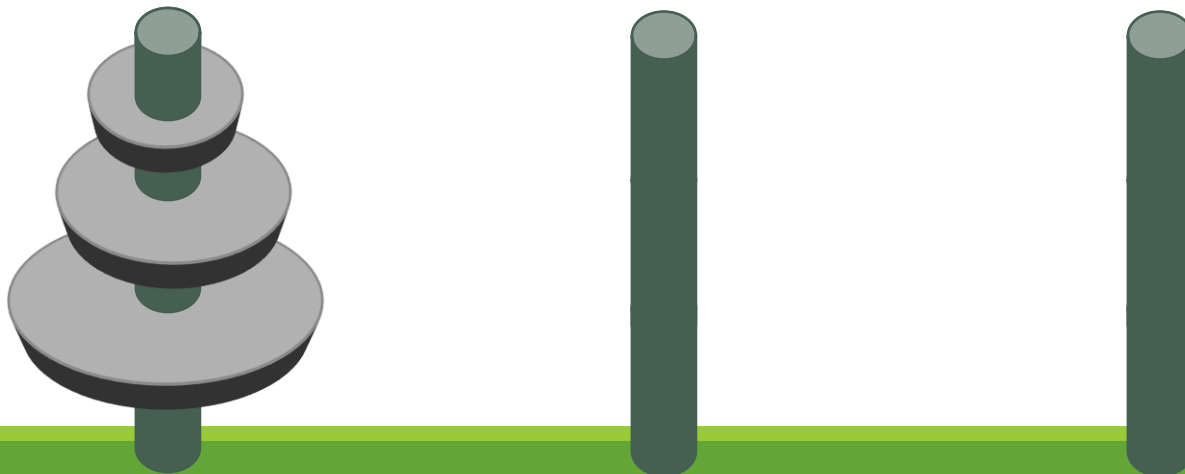
1. Bata a margarina, as gemas e o açúcar **por 15 minutos** à mão
2. Junte o leite e a farinha e bata por **5 minutos** à mão
3. Acrescente o fermento e, **por último**, as claras em neve
4. Unte a forma com manteiga e, **em seguida**, deposite a massa
5. Leve ao forno para assar **por 30 minutos, forno a 180°C**

Qualquer pessoa já pode fazer o bolo?

Torre de Hanói

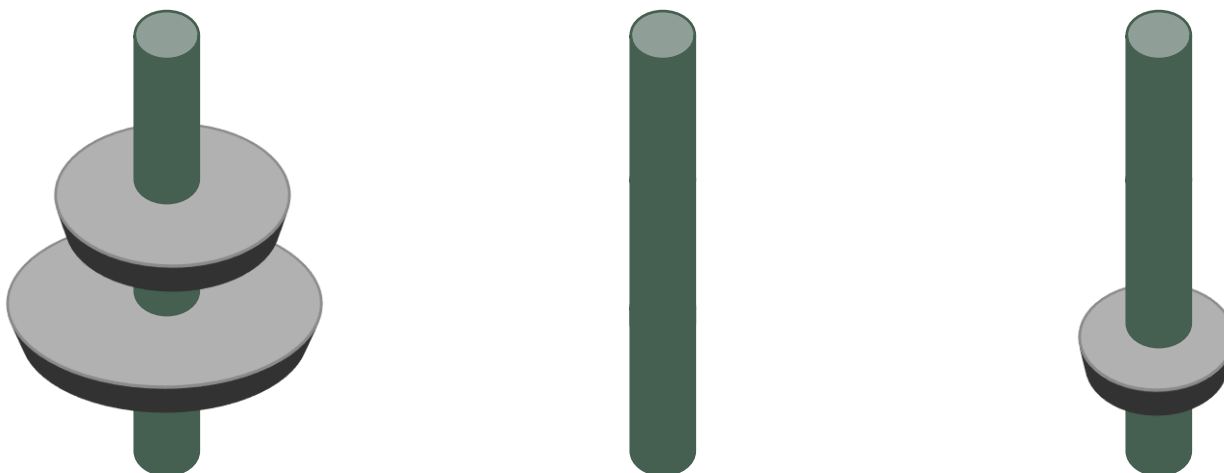
Escrevendo um algoritmo informal para resolver o problema da Torre de Hanói. Atente que, na Torre de Hanói:

- 1. Deve-se mover todos os discos do *primeiro* eixo para o *terceiro* mantendo-se a ordem original;**
- 2. Em *cada* movimento, pode-se mover apenas *um* disco;**
- 3. Um disco nunca poderá ser *sobreposto* por outro *maior*.**



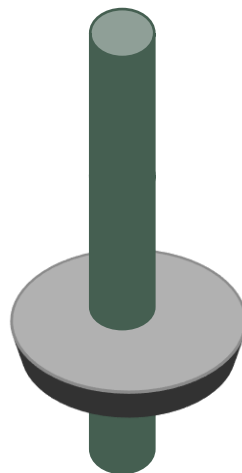
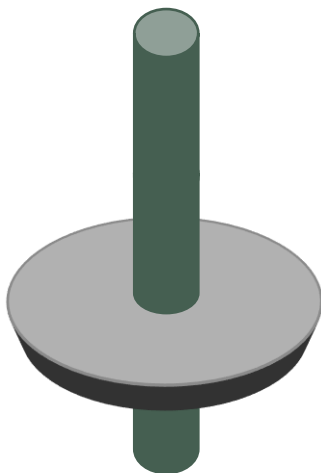
Passo 1

1. Mova o disco **MENOR** para o **TERCEIRO** eixo



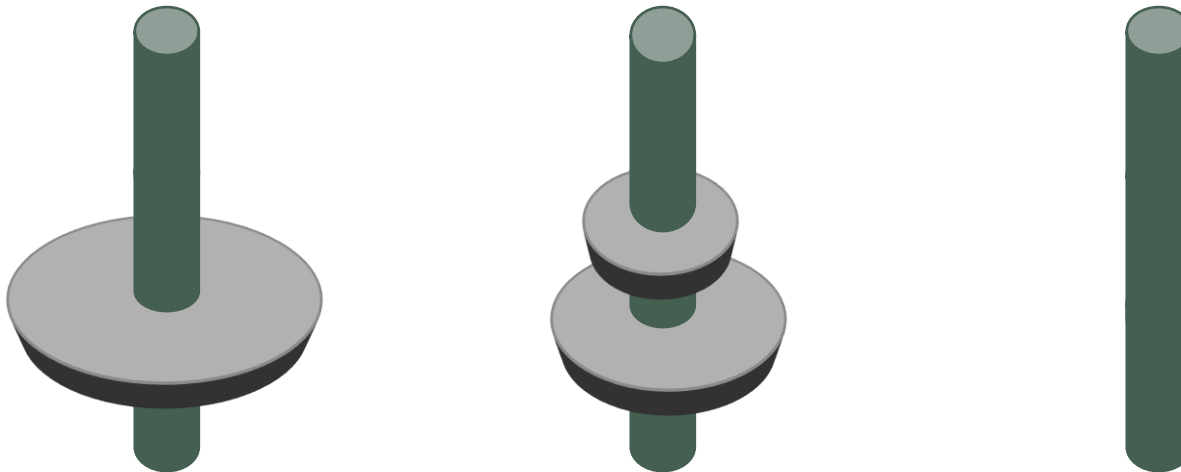
Passo 2

2. Mova o disco MÉDIO para o SEGUNDO eixo



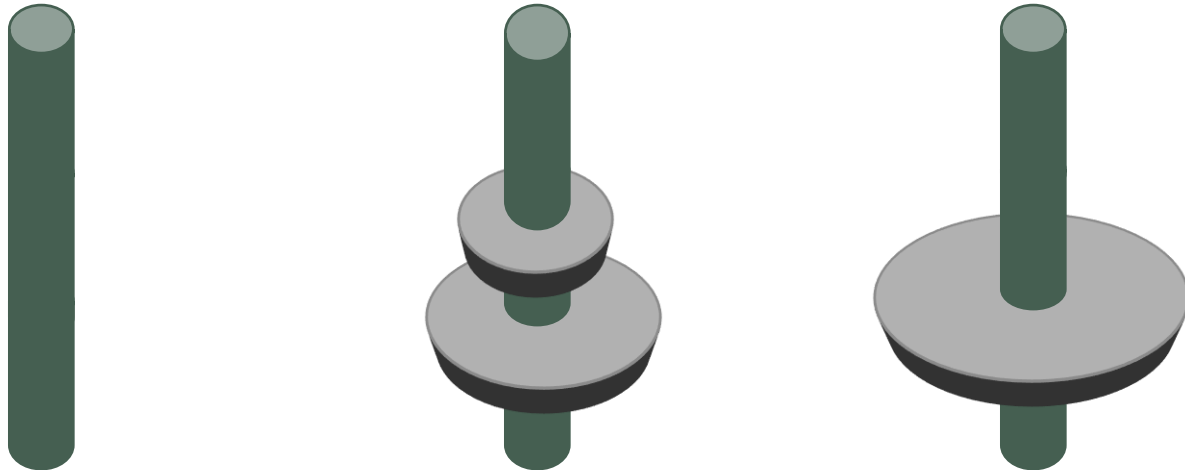
Passo 3

3. Mova o disco **MENOR** para o **SEGUNDO** eixo



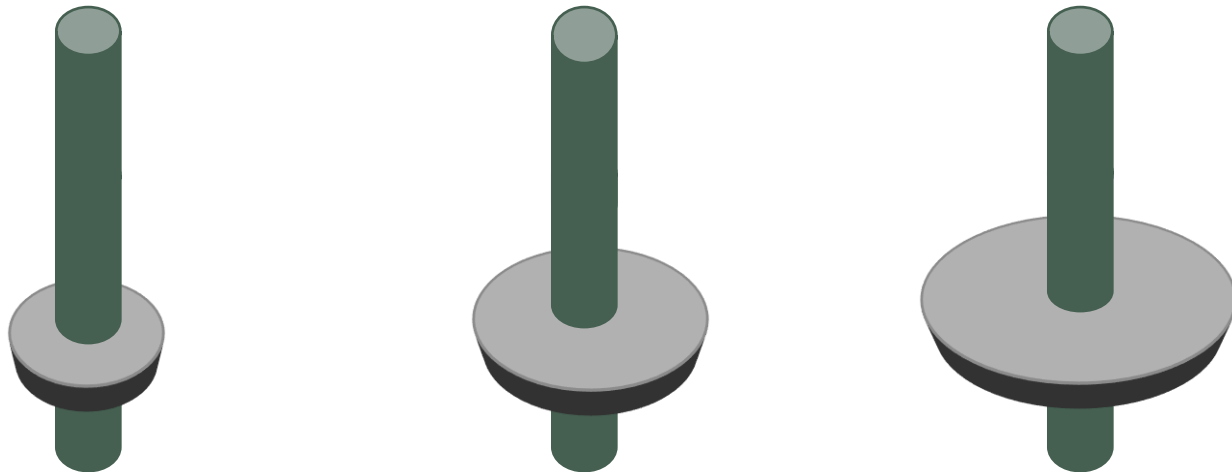
Passo 4

4. Mova o disco **MAIOR** para o **TERCEIRO** eixo



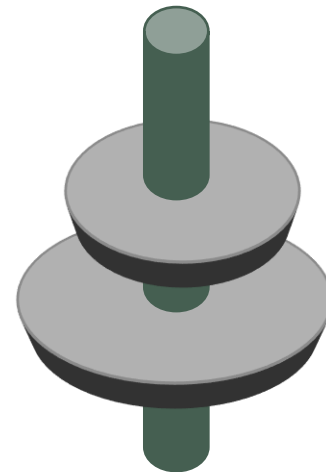
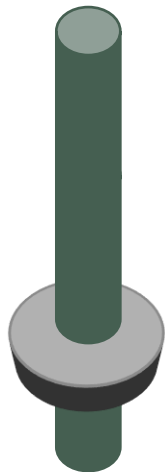
Passo 5

5. Mova o disco **MENOR** para o **PRIMEIRO** eixo



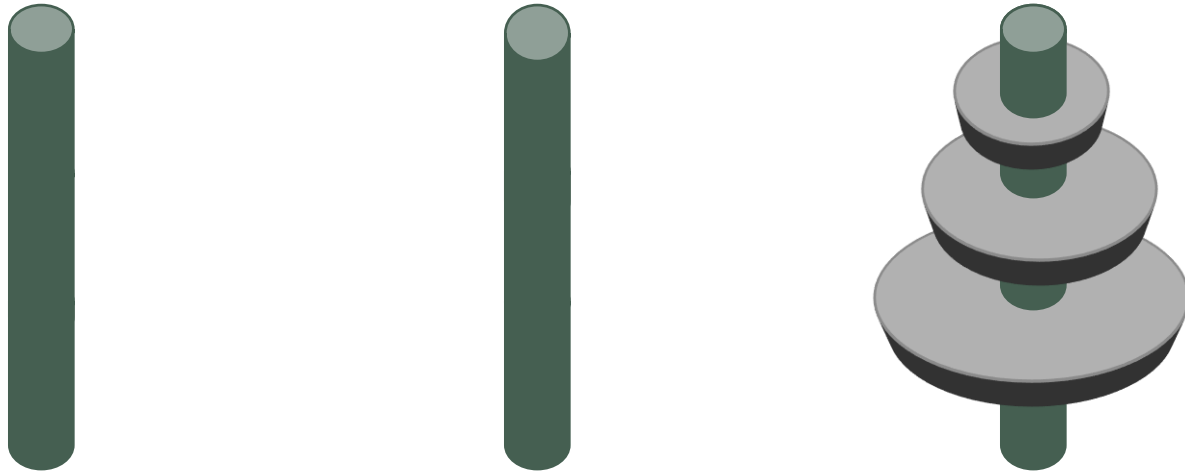
Passo 6

6. Mova o disco MÉDIO para o TERCEIRO eixo



Passo 7

7. Mova o disco MENOR para o TERCEIRO eixo



Algoritmo

Início

- 1. Mova o disco MENOR para o TERCEIRO eixo**
- 2. Mova o disco MÉDIO para o SEGUNDO eixo**
- 3. Mova o disco MENOR para o SEGUNDO eixo**
- 4. Mova o disco MAIOR para o TERCEIRO eixo**
- 5. Mova o disco MENOR para o PRIMEIRO eixo**
- 6. Mova o disco MÉDIO para o TERCEIRO eixo**
- 7. Mova o disco MENOR para o TERCEIRO eixo**

Fim

Formas de Representação

1. Descrição Narrativa

Utilização do português para descrição do algoritmo

2. Fluxograma convencional

Utilização de símbolos gráficos na representação do algoritmo

3. Linguagem Algorítmica (Pseudocódigo)

Utilização de uma pseudo-linguagem de programação
("português estruturado" ou "portugol")

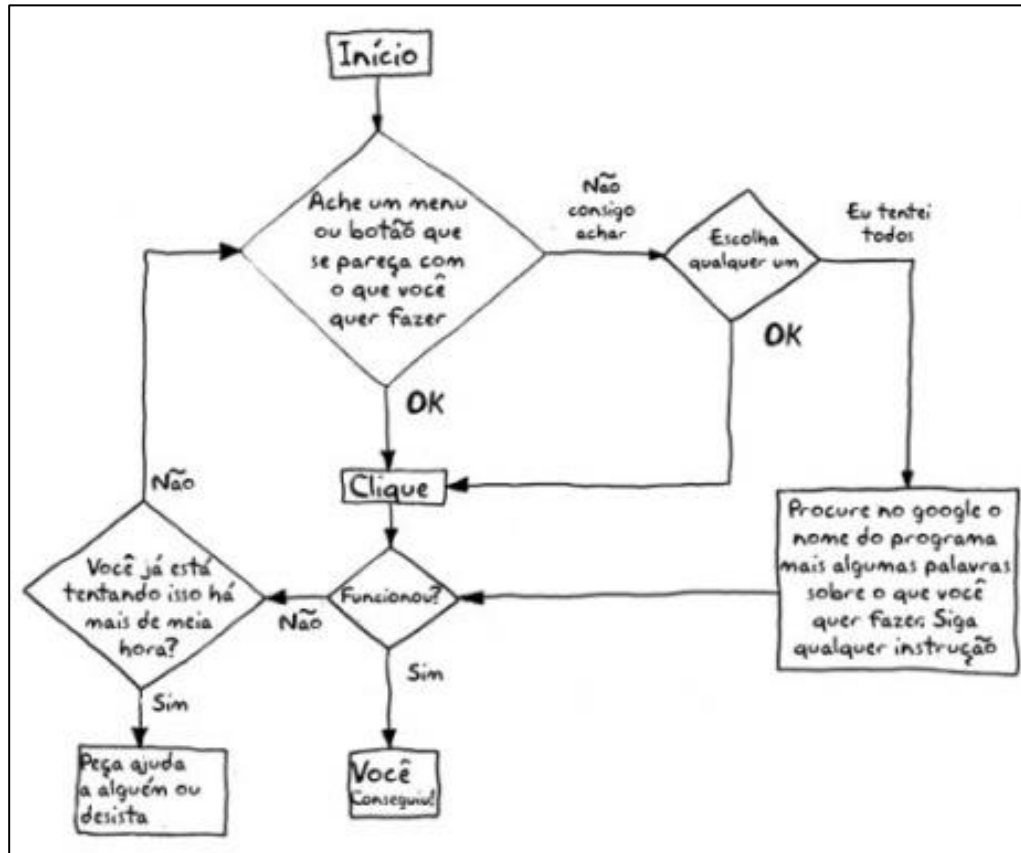
Descrição Narrativa - Exemplo

Algoritmo para Trocar uma Lâmpada

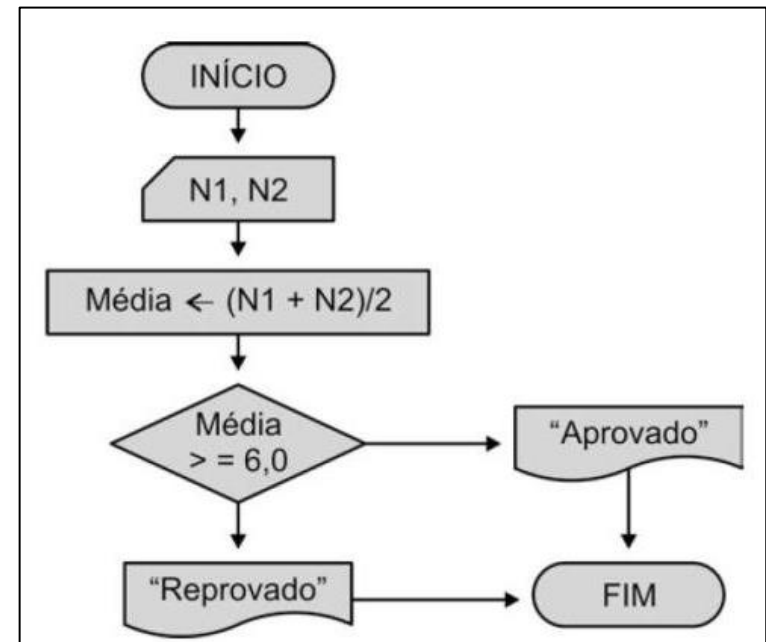
```
Início
  Verifica se o interruptor está desligado;
  Procura uma lâmpada nova;
  Pega uma escada;
  Leva a escada até o local;
  Posiciona a escada;
  Sobe os degraus;
  Para na altura apropriada;
  Retira a lâmpada queimada;
  Coloca a lâmpada nova;
  Desce da escada;
  Aciona o interruptor;
    Se a lâmpada não acender, então:
      Retira a lâmpada queimada;
      Coloca outra lâmpada nova
    Senão
      Tarefa terminada;
  Joga a lâmpada queimada no lixo;
  Guarda a escada;
Fim
```

Fluxograma - Exemplos

Algoritmo para orientar um usuário de computador



Algoritmo p/ verificar a situação de um aluno



Linguagem Algorítmica - Exemplo

Algoritmo para verificar a situação de um aluno

Algoritmo PROGRAMA_EXEMPLO

Variaveis

A, B, MEDIA: REAIS;

Inicio

ESCREVA ("Digite a nota da primeira prova:");

LEIA (A);

ESCREVA("Digite a nota da segunda prova:");

LEIA (B);

$MEDIA \leftarrow (A + B) / 2;$

ESCREVA ("A MEDIA EH: ", MEDIA);

SE (MEDIA \geq 60) ENTAO

 ESCREVA ("APROVADO!");

SENAO

 ESCREVA ("REPROVADO!");

FIM_SE

Fim

Fazendo uso das Formas de Representação

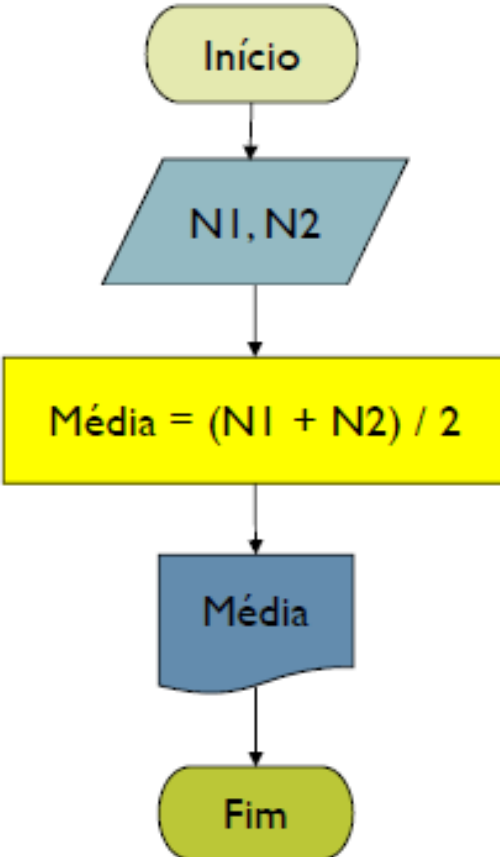
- Considere o seguinte problema...

Faça um algoritmo que receba dois numeros reais, calcule e imprima a média aritmética entre eles.





Como representar uma solução para este problema



Comparativo das Formas de Representação

Desc. Narrativa	Fluxograma	Linguagem Algorítmica
<p>Solicite que alguém informe dois números. Adicione estes dois números. Faça o cálculo da média aritmética e informe o resultado</p>	 <pre> graph TD Inicio([Início]) --> Entrada[/N1, N2/] Entrada --> Processa[Média = (N1 + N2) / 2] Processa --> Saida[/Média/] Saida --> Fim([Fim]) </pre>	<p>Algoritmo Calcula_Media N1, N2, M: real; Início escreva ("Digite Nota1 e Nota2"); leia (N1, N2); $M \leftarrow (N1 + N2) / 2$; escreva ("Média = ", M); Fim</p>

Comparativo das Formas de Representação

Desc. Narrativa	Fluxograma	Linguagem Algorítmica
<div>   </div> <div>Vantagens</div>		
<ul style="list-style-type: none"> ▪ O português é conhecido 	<ul style="list-style-type: none"> ▪ Figuras dizem muito mais que palavras; ▪ Padrão mundial 	<ul style="list-style-type: none"> ▪ Usa o português como base; ▪ Pode-se definir quais e como os dados vão estar estruturados;
<div>   </div> <div>Desvantagens</div>		
<ul style="list-style-type: none"> ▪ Imprecisão; ▪ Baixa confiabilidade ▪ Em algumas situações, escreve-se muito 	<ul style="list-style-type: none"> ▪ Pouca atenção aos dados, quanto a descrição; ▪ Difícil representar à medida que o algoritmo cresce. 	<ul style="list-style-type: none"> ▪ Exige a definição de uma linguagem não real para trabalho; ▪ Não padronizado.



Linguagem Algorítmica - Exemplos

Exemplo 1 – Mensagem de boas vindas

```
algoritmo mensagem_boas_vindas  
inicio  
    escreva ("Olá usuário, seja bem-vindo ao mundo dos algoritmos! ");  
fim
```

Linguagem Algorítmica - Exemplos

Exemplo 2 – Soma de dois números

Faça um algoritmo que leia dois números inteiros, some-os e mostre o resultado

```
algoritmo soma_numeros
    numero1, numero2: inteiro;
inicio
    escreva ("Forneça dois números: ");
    leia (numero1, numero2);
    escreva ("A soma dos números é: ", numero1 + numero2);
fim
```

Linguagem Algorítmica - Exemplos

Exemplo 3 – Dobro de um número

Faça um algoritmo que leia um número real, calcule o seu dobro e mostre o resultado

```
algoritmo calcula_dobro
    numero, dobro: real;
inicio
    escreva ("Forneça um número: ");
    leia (numero);
    dobro ← numero * 2;
    escreva ("O valor do dobro é: ", dobro);
fim
```

Elementos Básicos da Linguagem Algorítmica

- Tipos de dados
- Operadores e expressões
- Variável
- Operação de atribuição
- Operações de entrada e saída
- Estruturas de controle:
 - Sequenciação
 - Decisão
 - Repetição

■ Sequenciação

- Sequência linear de instruções, executadas uma após a outra.
- Os comandos do algoritmo fazem parte de uma sequência, onde é relevante a ordem na qual se encontram os mesmos.

Comando-1
Comando-2
Comando-3
:
Comando-n

Estruturas de Controle

■ **Decisão** (Condicional)

- Há a subordinação da execução de um ou mais comandos à veracidade de uma condição.
- Exemplos:
 - Se tiver dinheiro suficiente, então vou almoçar em um bom restaurante.
 - Caso contrário (senão), vou comer um sanduíche na lanchonete da esquina.

Estruturas de Controle

■ **Repetição** (Laço ou “Loop”)

- Permite que tarefas individuais sejam repetidas um número determinado de vezes ou tantas vezes quantas uma condição lógica permita.
- Exemplos:
 - Vou atirar pedras na vidraça até quebrá-la;
 - Baterei cinco pênaltis;
 - Enquanto tiver saúde e dinheiro, vou desfrutar a vida.

Programa

- É um algoritmo codificado em alguma linguagem de programação
- Exemplo em Python:

Programa para calcular o dobro de um número inteiro

```
num = int(input('Digite um número inteiro: '))  
  
dobro = num * 2  
  
print('O dobro de', num, 'é', dobro)
```

Como o computador entende as minhas instruções?

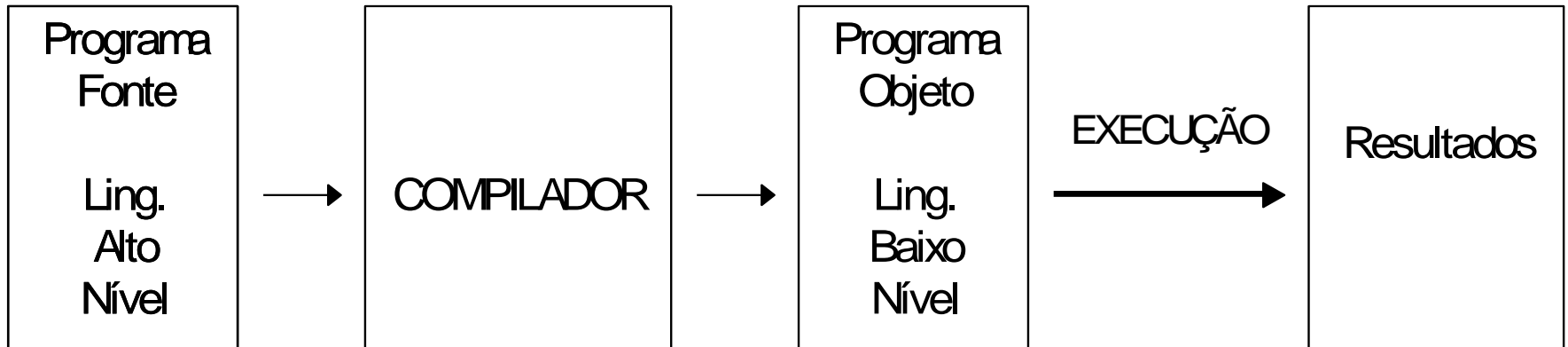
Linguagens de programação precisam ser **transformadas** em **instruções de máquina** para que possam ser executadas

Compiladores e Interpretadores

- são programas responsáveis pela **tradução** das linguagens de programação

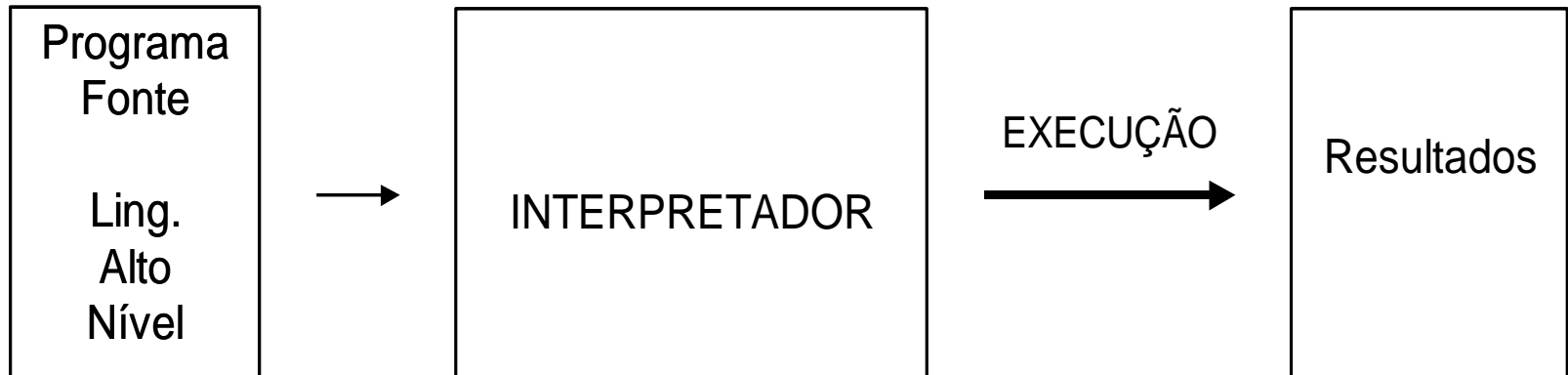
O que é um Compilador

COMPILADOR - traduz o programa-fonte para um programa equivalente escrito em linguagem de máquina (programa-objeto).



O que é um Interpretador

INTERPRETADOR - traduz e envia para execução, instrução por instrução e o programa permanece na forma fonte.



Compilador e Interpretador

No processo de

Compilação é preciso fazer a tradução do código fonte da nossa aplicação para cada plataforma destinada (como versões específicas de uma aplicação para Windows, Linux e Mac OS).

Interpretação, por fazer a tradução em tempo de execução, o processo independe da plataforma

- Pode ser utilizada uma **metodologia híbrida**: compilador + interpretador (por exemplo, Python faz compilação e interpretação de **bytecode** em uma máquina virtual **Python**)

Próxima aula ...

Programando em Python