# Linguagem e Técnica de Programação 1

**CAPÍTULO – 1**

## Introdução à Lógica de Programação

**Prof. Arnaldo**

# SUMÁRIO

- **Considerações iniciais**
  - Lógica
  - Raciocínio lógico
  - Argumentos
    - Dedutivos
    - Indutivos.
- **Algoritmo**
  - Fases, Estrutura e Teste de mesa.
- **Forma de representação gráfica**
  - Fluxograma
- **Programas**
  - Compilador e Interpretador.

# Considerações Iniciais

- O filósofo grego **Aristóteles** é considerado o **criador da lógica**.

- A palavra “lógica” é originária do grego **logos**, que significa linguagem racional.

- De acordo com o **dicionário Michaelis**, **lógica** é a análise das formas e leis do pensamento, mas não se preocupa com a produção do pensamento, mas sim com sua forma, isto é, com a maneira pela qual um pensamento ou uma **ideia** é **organizada** e **apresentada**, **possibilitando** que cheguemos a uma **conclusão**.

# Considerações Iniciais

- **A lógica** está presente em nossa vida sempre que pensamos, falamos e escrevemos, pois realizar ações necessitamos que os pensamentos estejam ordenados de modo a alcançar o resultado esperado.

- **O uso da lógica** é de extrema importância para os profissionais de informática, seu papel dentro das empresas é solucionar problemas e atingir os objetivos esperados pelos usuários, desse modo ela auxilia a análise desses problemas de modo a solucioná-los da forma mais rápida possível.

- **Na programação**, a lógica possui a mesma função, ordenar as instruções em uma sequência lógica para atingir um objetivo.

# Considerações Iniciais

- Antes de construir um programa utilizando uma linguagem de programação qualquer, o **programador** deve **montar** um **algoritmo** demonstrando seu **raciocínio lógico** sobre o problema em questão.

- A importância do algoritmo é a lógica utilizada por ele, se é **consistente** e **sem duplo sentido**.

- Depois que a lógica da programação estiver pronta pode-se **implementar** em qualquer **linguagem de computador**.

- **Raciocínio Lógico** é um modo de pensar que ajuda a resolver um problema ou chegar a uma conclusão sobre determinado assunto.

# Considerações Iniciais

- O uso do **raciocínio lógico** não é exclusivo dos profissionais da informática, podemos identificar a utilização da lógica no nosso dia-a-dia quando analisamos situações como:

  - **Todo mamífero bebe leite.**
  - **O homem bebe leite**.

  Portanto, **conclui-se** que:
  - **O homem é um mamífero**.

  A situação apresentada é uma argumentação **dedutiva** ou **indutiva**?

  **A seguir**

# Argumento

- Um **argumento** pode ser composto por uma ou várias premissas, as quais podem ser verdadeiras ou falsas e conduzem à conclusão, que também poderá ser verdadeira ou falsa.

Exemplo:
Temos em 1 e 2 as **premissas** e em 3 a **conclusão**.

1. **Sandra é mais velha do que Ricardo.**
2. **Ricardo é mais velho do que Pedro**.
3. **Logo, Sandra é mais velha do que Pedro.**

# Argumento: indutivo

- Os argumentos podem ser **dedutivos** ou **indutivos**.

- **Indutivos** são aqueles que, com base em dados, chega-se a uma resposta por meio de **analogia**, ou seja, pela comparação com algo conhecido. Esse tipo de raciocínio, contudo, **não oferece certeza** de que a **resposta** será de fato **verdadeira**. É necessário conhecer os fatos ou as situações para que se possa fazer a comparação. Por exemplo:

1. **Ontem não havia nuvens no céu e não choveu.**
2. **Hoje não há nuvens no céu**.
3. **Portanto, hoje não vai chover.**

# Argumento: indutivo

- **Na argumentação indutiva,**
  - ✓ os casos singulares são elevados ao universal;
    - ❑ no caso do exemplo anterior, o caso de "**ontem não havia nuvem no céu**" (**premissa 1**) comparando ao caso de "**hoje também não há nuvens no céu**" (**premissa2** ) conduziu a uma **conclusão** (**induzida**).

# Argumento: dedutivo

- **Dedutivos** são cuja a conclusão é obtida como consequência das premissas, isto é, por meio da análise das situações ou fatos pode-se obter a resposta. Trabalha-se com a forma das sentenças, sem que haja necessidade do conhecimento prévio das situações ou fatos, isto é, a conclusão é obtida em decorrência das premissas. Por exemplo:

1. **Joana é uma mulher.**
2. **As mulheres são seres humanos**.
3. **Logo, Joana é um ser humano.**

# Algoritmo

**Algoritmo** é uma sequência lógica de passos que levam a um determinado objetivo.

Exemplo:

**Trocar uma lâmpada**:

1) Pegue uma escada;
2) Coloque-a embaixo da lâmpada velha;
3) Pegue uma lâmpada nova;
4) Suba na escada;
5) Retire a lâmpada velha;
6) Coloque a lâmpada nova;
7) Desça da escada.

A elaboração de um **algoritmo** para a **criação de um programa** de computador requer algumas etapas:

1. Definir o problema;
2. Estudar a situação atual e analisar a forma de resolver o problema;
3. Desenvolver o programa utilizando uma linguagem de programação;
4. Após a implementação, analisar junto aos usuários se o problema foi resolvido.

# Algoritmo: fases

O problema que o **algoritmo** representa é composto por **três fases**:


Sendo que:

- Entrada: são dados de entrada de um algoritmo.
- Processamento: são os procedimentos utilizados para chegar ao resultado final.
- Saída: são os dados já processados.

# Algoritmo: estrutura

Os algoritmos são construídos em uma linguagem chamada **pseudocódigo** seguindo uma estrutura básica para qualquer linguagem de programação.

Para exemplificar os códigos, utilizaremos um pseudocódigo chamado **portugol**, e sua equivalência em fluxograma.

Estrutura básica para os algoritmos:

```
algoritmo <nome do programa>
<definições>
var
<variáveis>
inicio
<comandos>
fimalgoritmo
```

**Observação 1:** *Pseudocódigo significa que o algoritmo está sendo desenvolvido em uma linguagem que não é uma linguagem de programação.*

**Observação 2:** *Portugol não é uma linguagem de programação, pois não existe um compilador que execute seus comandos dentro do computador, servindo apenas para auxiliar o programador a montar uma estrutura de programa.*

# Algoritmo: teste de mesa

O teste de mesa é nada mais é do que seguir as instruções de algoritmo de maneira precisa para verificar se há erro na sua estrutura, independente da linguagem em que será implementado.

**Exemplo:**

Algoritmo de calculo da média final considerando a nota da prova 1 como 7,0 e da prova 2 como 9,0.

1. Receba a nota da prova 1;
2. Receba a nota da prova 2;
3. Some as notas e divida o resultado por 2;
4. Exiba o resultado da divisão.

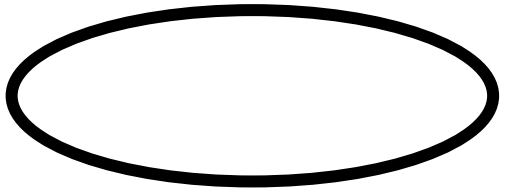
| Prova 1 | Prova 2 | Média |
|---|---|---|
| 7,0 | 9,0 | 8,0 |

# Formas de Representação Gráfica: fluxograma

- Os algoritmos pode ser representados por um conjunto de símbolos padronizados peara facilitar a compreensão do código, chamado **fluxograma**.
- O **fluxograma** é uma ferramenta usada e desenvolvida pelos profissionais da análise de sistemas, e tem como finalidade descrever o fluxo das informações.
- As figuras são formas geométricas básicas que representam a entrada, processamento e saída de dados.

# Formas de Representação Gráfica: fluxograma

Alguns dos símbolos mais conhecidos e utilizados:



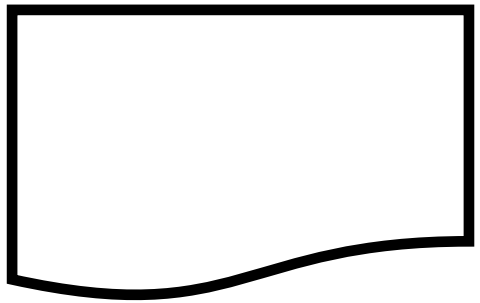
**Terminal:** utilizado para indicar início ou fim de algoritmo.

**Seta de fluxo de dados:** indica o sentido do fluxo de dados.

**Processamento:** manipulação de dados.

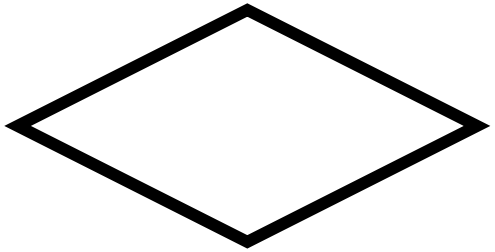
# Alguns dos símbolos mais conhecidos e utilizados:

**Entrada:** dispositivo qualquer de entrada dados.



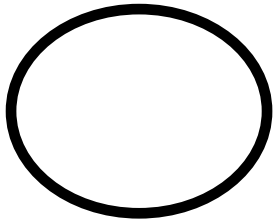
**Saída de dados em impressora:** representa os dados que serão impressos.

**Saída de dados em vídeo:** representa os dados que serão exibidos na tela.

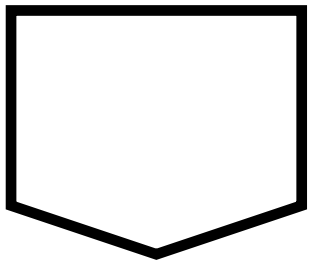


**Decisão:** indica a decisão que deve ser tomada, mostrando a possibilidade de desvios para outros pontos do fluxo dependendo do resultado da comparação

# Alguns dos símbolos mais conhecidos e utilizados:



**Conector:** é utilizado quando é preciso dividir o fluxograma.

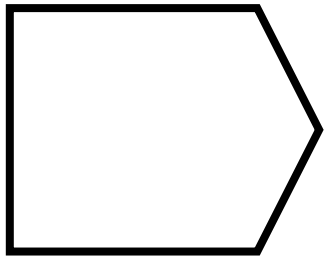


**Conector de páginas:** específico para indicar conexão do fluxo em outra página.

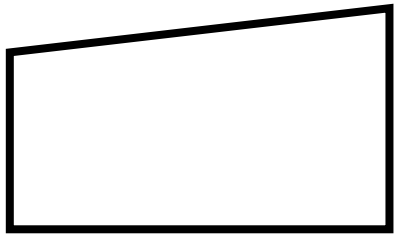
**Preparação ou Processamento predefinido:** representa um bloco de operações que não estão incluídas na diagramação.

**Sub-rotina:** representa um trecho de instruções que está fora do programa principal.

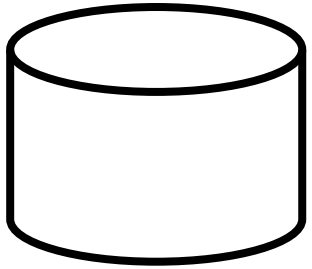
# Outros símbolos:



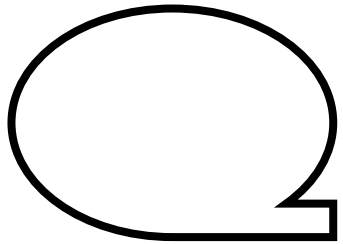
**Modificação de programa:** indica a existência de uma instrução ou de um grupo de instruções que irão modificar o programa.



**Teclado:** são as informações recebidas ou fornecidas.



**Disco Magnético:** memória de massa para armazenamento de dados.

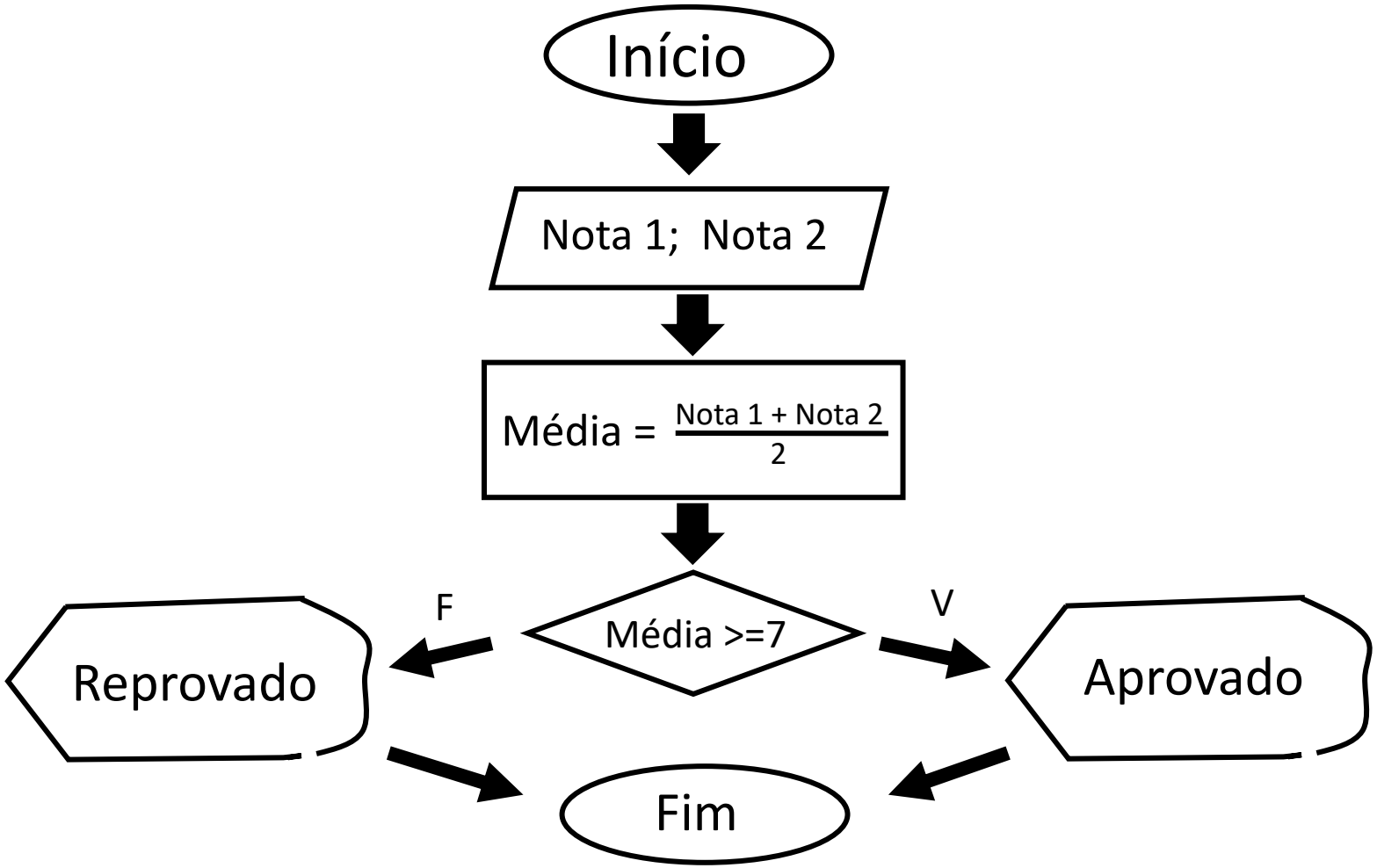


**Fita Magnética:** memória de massa para armazenamento de dados.

# Fluxograma

**Exemplo:**

- um algoritmo para a verificação da aprovação ou reprovação de um aluno.



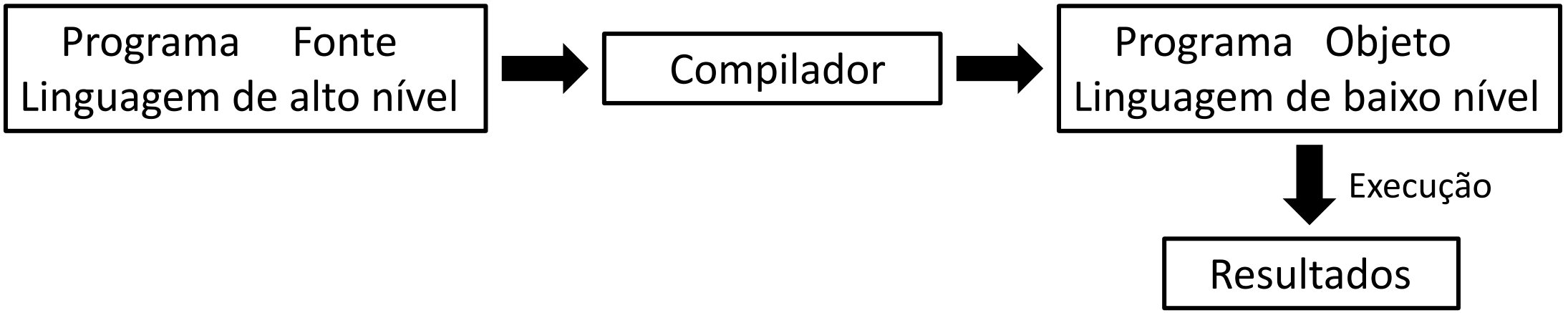
# Programas

Os **programas de computador** são um conjunto de algoritmos escritos em uma linguagem que o computador consegue interpretar, como: Java, C, C++, C#, Python, PHP, Visual Basic.Net, Perl, JavaScript, Delphi/Object Pascal, Ruby, Visual Basic, Assembly Language, Objective-C, D, Swift, R, MATLAB, PL/SQL, Groovy (essas são as **linguagens de programação mais usadas** em 2016).

Para que o computador possa entender o que está sendo pedido pelo programa é necessário um meio de tradução entre a linguagem utilizada pelo programa e a linguagem de máquina. Esta tradução é feita pelo **compilador** ou pelo **interpretador**.

# Programa: compilador

**Compilador** é um **programa** que transforma o código escrito em uma linguagem de programação, em um programa equivalente em outra linguagem, **código objeto**. O código objeto é escrito em uma **linguagem de baixo nível**, como uma sequência de instruções a ser executada por um sistema computacional.




O processo de compilação é composto de análise e síntese, onde a **análise** tem como objetivo entender o código fonte e representá-lo em uma estrutura intermediária, e a **síntese** constrói o código objeto a partir dessa representação intermediária.

# Programa: compilador

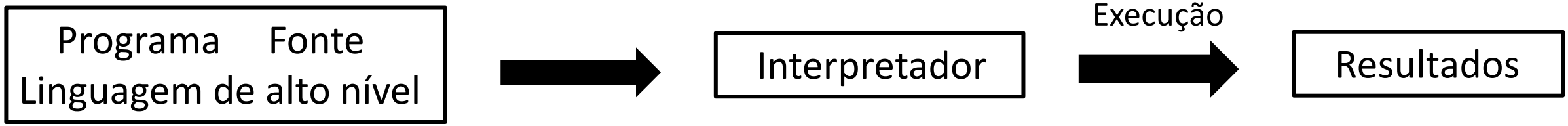
- **Vantagens:**
  - O código compilado é mais rápido de ser acessado;
  - Impossibilita ou pelo menos dificulta ser quebrado e visualizado o código-fonte original;
  - Permite otimização do código por parte do compilador;
  - Compila o código somente se estiver sem algum erro.

- **Desvantagens:**
  - Para ser utilizado o código precisa passar por muitos níveis de compilação;
  - Assim como vantagem a possibilidade de não poder visualizar o código-fonte, pode ser uma desvantagem;
  - Processo de correção ou alteração do código requer que ele seja novamente recompilado.

# Programa: interpretador

**Interpretadores** são programas que leem um **código fonte** de uma **linguagem de programação** e os convertem em código executável.

Linguagens interpretadas são mais **dinâmicas** por não precisarem escrever-compilar-testar-corrigir-compilar-testar-distribuir, e sim escrever-testar-corrigir-escrever-testar-distribuir. Mas existem também linguagens que funcionam como interpretadores e **compiladores** como o C.




A princípio, qualquer linguagem de programação pode utilizar compiladores e interpretadores. Mas para determinadas linguagens é mais fácil desenvolver interpretadores, e para outras é mais prático um compilador.

# Programa: compilador

- **Vantagens:**
  - Correções e alterações são mais rápidas de serem realizadas;
  - Código não precisa ser compilado para ser executado;
  - Consomem menos memória.

- **Desvantagens:**
  - Execução é mais lenta do programa;
  - Necessita sempre ser lido o código original para ser executado;

# Referencias Bibliográficas


- PUGA, Sandra e RISSETTI, Gerson. Lógica de Programação Estruturas de Dados, com aplicações em Java. 2.ed. São Paulo: Pearson Prentice Hall, 2009.

- SILVA, Camila Ceccatto da e PAULA, Everaldo Antônio de. Lógica de Programação: aprendendo a programar. Santa Cruz do Rio Pardo, SP: Viena, 2007.

- Disponível em: OFICINA DA NET. Diferenças entre compiladores e interpretadores. https://www.oficinadanet.com.br/artigo/1527/diferencas_entre_compiladores_e_interpretadores. Acessado em: 04/06/2017.