Programa e Linguagem de Programação

Disciplina: Algoritmos e Programação
Curso: Engenharia de Computação

Professora: Mariza Miola Dosciatti
mariza@utfpr.edu.br

Objetivos

- Entender o conceito de linguagem de programação e lógica de programação.
- Compreender os tipos de linguagens de programação e formas de diferenciá-las.
- Entender o processo de compilação e de interpretação.
- Compreender os paradigmas de programação e suas caracterizações básicas.

- Item da ementa (Plano de Ensino):
  - Conceito de algoritmo e programação.

Sumário

Antes de começar, alguns conceitos...

- **O que é um problema?**

  Tema cuja solução ou resposta requer considerável pesquisa, estudo e reflexão.

  Dificuldade ou obstáculo que requer esforço para ser solucionado.

  Situação conflitante; dificuldade.

  …

  Fonte: Dicionário Michaelis

Problemas

- Fazem parte do nosso cotidiano.
- Exemplo de problemas cotidianos:
  - Trocar a resistência de um chuveiro.
  - Definir onde almoçar.
- Sempre que nos deparamos com um problema, buscamos uma solução para o mesmo.

Exemplo de solução

- Para trocar a resistência de um chuveiro devemos:
  - Adquirir uma resistência nova.
  - Abrir o chuveiro.
  - Remover a resistência defeituosa.
  - Colocar a resistência nova.
  - Fechar o chuveiro.
  - Descartar a resistência defeituosa.

Lógica

- A lógica é que orienta os passos para a solução de um problema.

- A **Lógica** é o ramo da Filosofia e da Matemática que estuda os métodos que permitem **distinguir entre raciocínios válidos e não válidos**, determinando o processo que leva ao verdadeiro conhecimento.

Noções de Lógica

- Lógica é a ciência que estuda as formas de pensamento.
- A Lógica nos acompanha diariamente:
  - Um bebê sabe que precisa chorar para receber atenção.
  - Se um carro está com a seta esquerda ligada, significa que ele vai virar à esquerda.

Existe lógica no nosso dia a dia?

- Sempre que pensamos, a lógica ou a ilógica necessariamente nos acompanha.
- Quando falamos ou escrevemos, estamos expressando nosso pensamento, logo, precisamos usar a lógica nessas atividades.
- **Exemplos:**
  a. Todo mamífero é um animal.
     Todo cavalo é um mamífero.
     Portanto, todo cavalo é um animal.

  b. João é mais velho que Maria.
     Maria é mais velha que Antônio
     Portanto, João é mais velho que Antônio

Dedução lógica

Todo mamífero é um animal. → premissa
Todo cachorro é um mamífero. → premissa

Todo cachorro é um animal. → conclusão

- **Silogismo**
  - É um **modelo de raciocínio baseado na ideia da dedução**, composto por duas premissas que geram uma conclusão. Faz parte da **Lógica Proposicional**.

Dedução lógica (cont.)

- Silogismo válido
  - Exemplo:

Pernambuco é um estado do Brasil
Tiago reside em Pernambuco
Logo, Tiago reside no Brasil

- Silogismo inválido
  - Exemplo:

Existem biscoitos feitos de água e sal.
O mar é feito de água e sal.
Logo, o mar é um biscoito.

Dedução lógica (cont.)

Silogismo inválido

Dedução lógica (cont.)

Vamos pensar em novos exemplos:

?

??

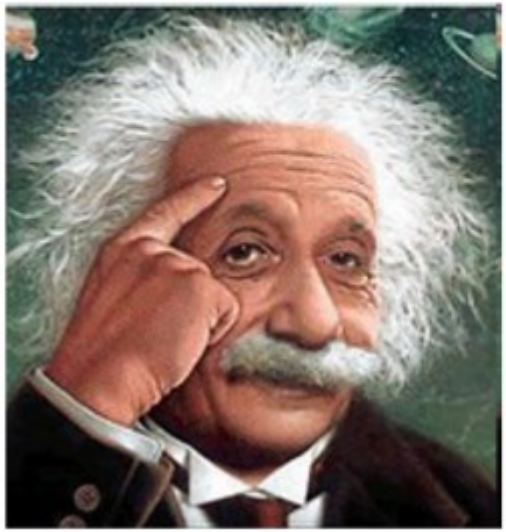
???

?????

??????

?????????

??????????



Resolva o problema a seguir:

Três senhoras - dona Branca, dona Rosa e dona Violeta – passeavam pelo parque quando dona Rosa disse:
- Não é curioso que estejamos usando vestidos de cores branca, rosa e violeta, embora nenhuma de nós esteja usando um vestido de cor igual ao seu próprio nome?
- Uma simples coincidência – respondeu a senhora com o vestido violeta.

Qual a cor do vestido de cada senhora?

Conclusão...

- **Dona Rosa** não veste cor rosa e cor violeta.
  - Logo, Dona Rosa veste cor branca.
- **Dona Violeta** não veste cor violeta e cor branca.
  - Logo, Dona Violeta veste cor rosa.
- **Dona Branca** não veste cor branca e cor rosa.
  - Logo, Dona Branca veste cor violeta.

Problema dos canibais e missionários

Usar um barco para atravessar três padres e três canibais de uma margem a outra do rio.

Restrição: Não deixar mais canibais do que padres em nenhuma das margens.

Solução...

travessia 1	missionário - canibal
travessia 2	missionário
travessia 3	canibal - canibal
travessia 4	canibal
travessia 5	missionário - missionário
travessia 6	missionário - canibal
travessia 7	missionário - missionário
travessia 8	canibal
travessia 9	canibal - canibal
travessia 10	canibal
travessia 11	canibal - canibal

http://rachacuca.com.br/jogos/missionarios-e-canibais/

Problema do lobo, da ovelha e da couve

Usar um barco para atravessar o lobo, a ovelha e a couve de uma margem a outra do rio.

Restrição: Não deixar o lobo sozinho com a ovelha e não deixar a ovelha sozinha com a couve.

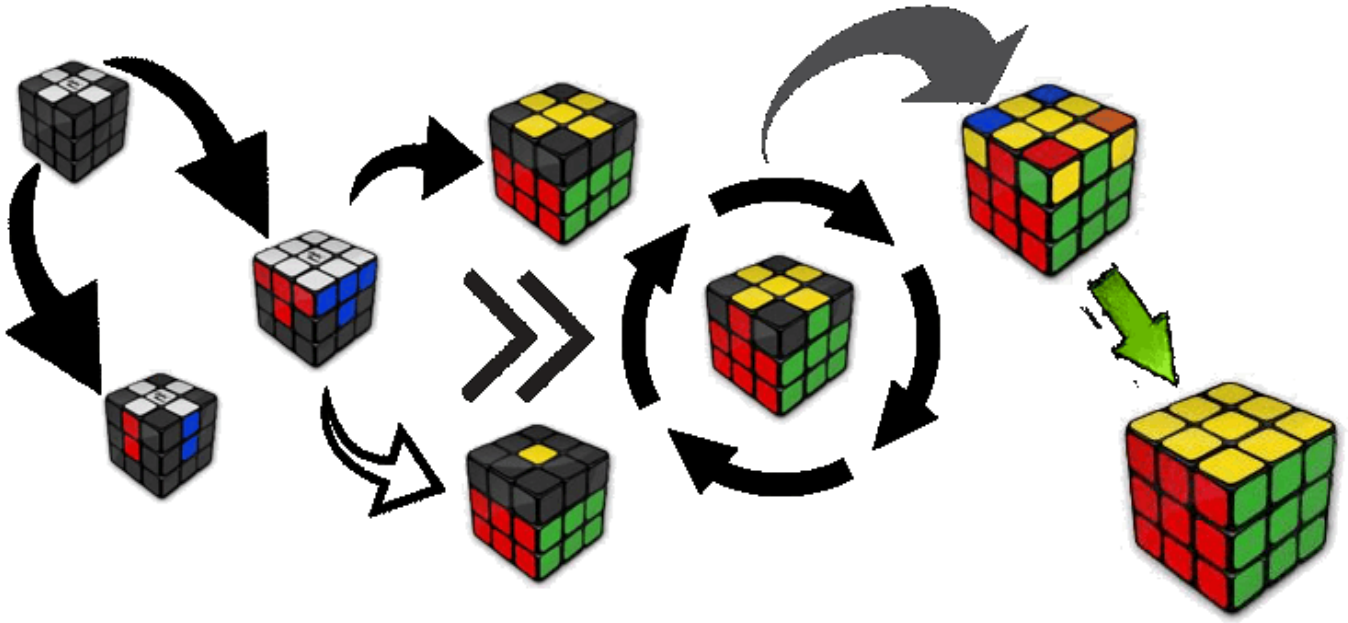
Solução...

travessia 1	Leva ovelha
travessia 2	Volta sozinho
travessia 3	Leva lobo e traz ovelha
travessia 4	Deixa ovelha e leva couve
travessia 5	Volta sozinho
travessia 6	Leva ovelha

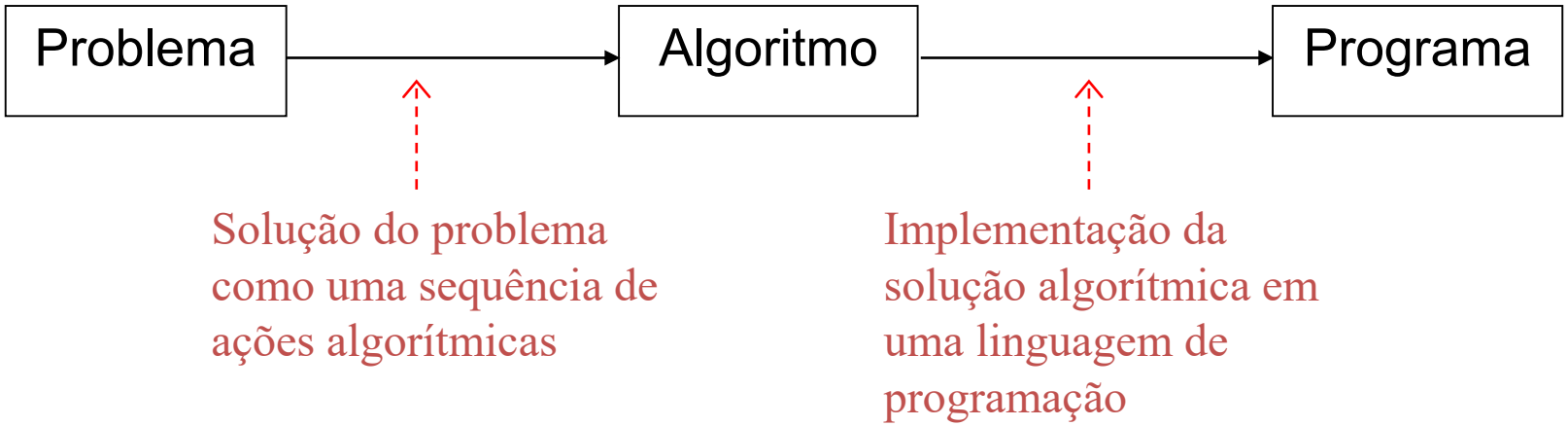
https://rachacuca.com.br/jogos/o-lobo-e-a-ovelha/

1. Lógica de programação

- **Lógica de programação**
  - É a técnica de desenvolver **sequências lógicas** para atingir um determinado objetivo.
  - As **sequências lógicas** são adaptadas para uma linguagem de computador.
  - Uma **sequência lógica** é denominada **algoritmo**.



2. Modelo


Problema
Algoritmo
Programa
Solução do problema como uma sequência de ações algorítmicas
Implementação da solução algorítmica em uma linguagem de programação

3. Algoritmo

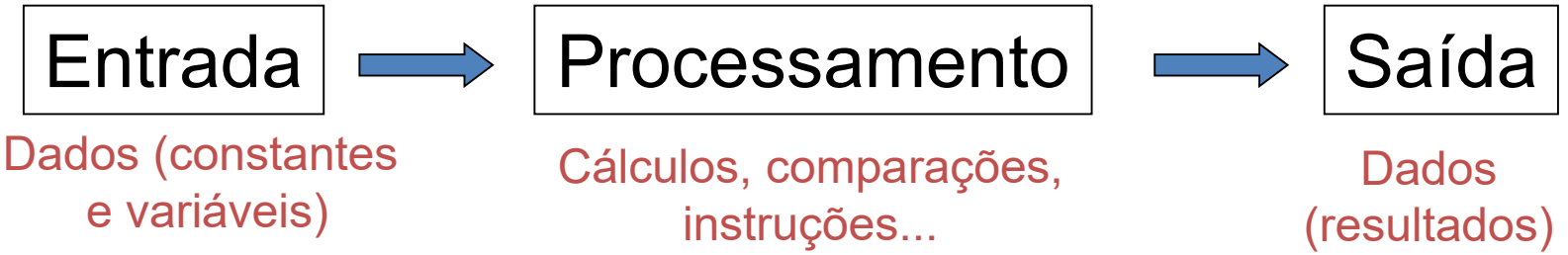
- **Um algoritmo é uma sequência finita de ações ou instruções** para executar uma tarefa, alcançar um objetivo ou obter uma saída desejada para quaisquer entradas válidas, **visando resolver um problema**.

4. Programa de computador

- **Algumas definições:**
  - Um programa de computador é um conjunto de instruções executadas em uma determinada sequência visando alcançar um objetivo.
  - Um programa de computador é a formalização de um algoritmo em uma linguagem inteligível pelo computador.
  - Um programa de computador é uma sequência de instruções que representam um algoritmo. Essas instruções são definidas de acordo com uma linguagem de programação, visando resolver problemas que possuem implementação computacional.

4. Programa de computador (cont.)

- Esquema de definição de um programa (solução algorítmica de um problema):



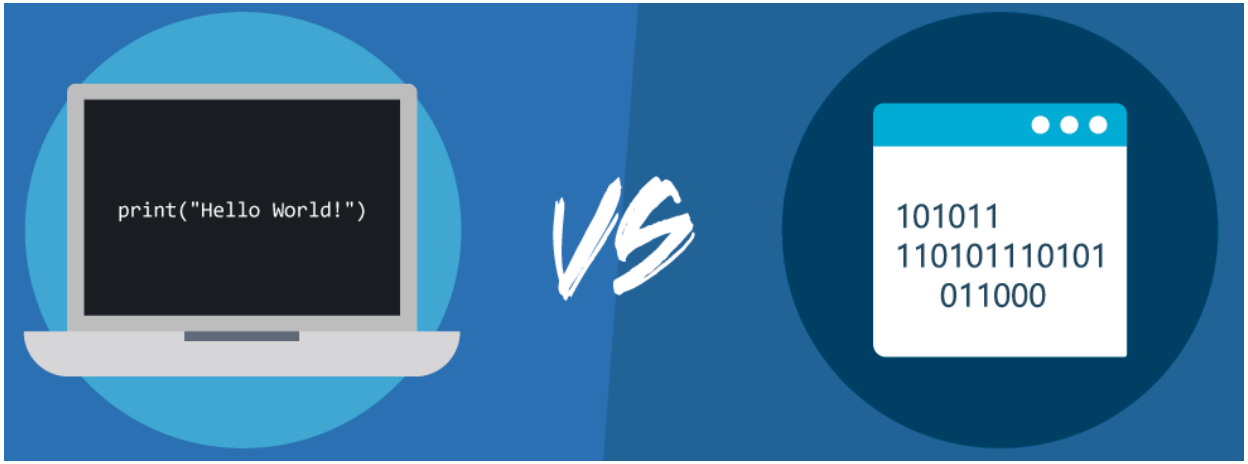
  - **Entrada**
    - Dados que serão utilizados no processamento
  - **Processamento**
    - Manipulação de variáveis e constantes
    - Resolução de expressões matemáticas
    - Decisão se determinadas instruções serão realizadas
    - Repetição de um conjunto de instruções de acordo com condições
    - Manipulação de dados em bases de dados e arquivos
    - ...
  - **Saída**
    - Resultados de processamento

5. Linguagens de programação

- Uma **linguagem de programação** define o conjunto de símbolos e as regras para expressar instruções computacionais.

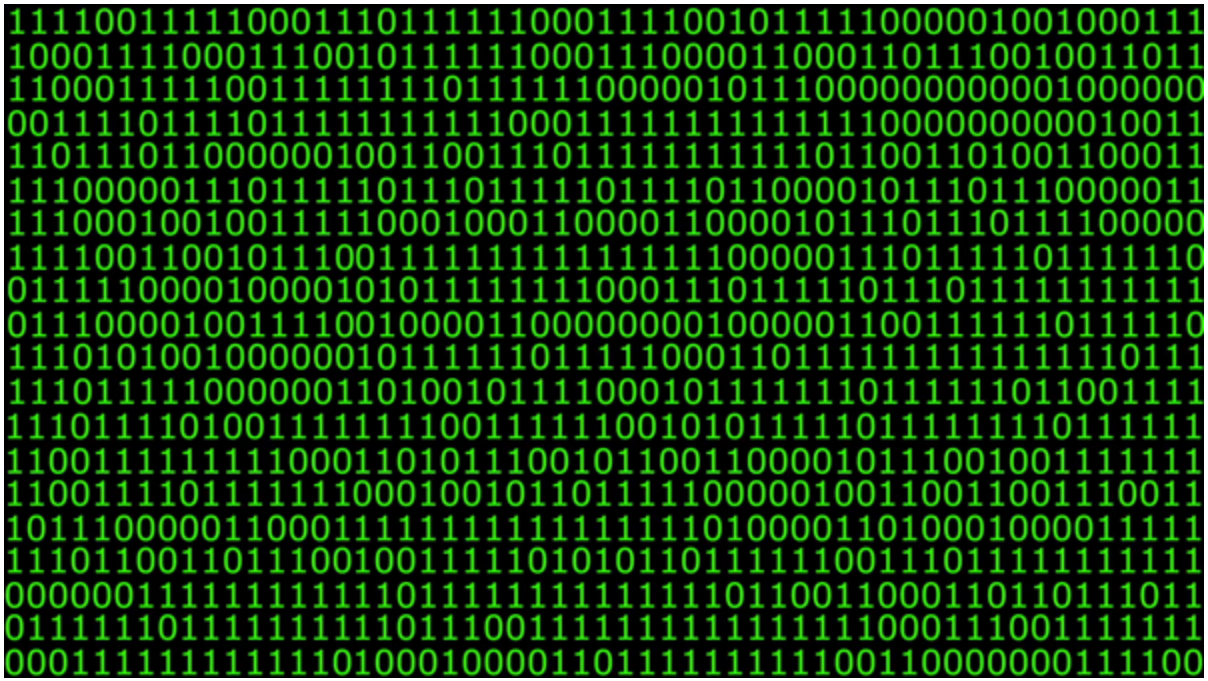
5. Linguagens de programação (cont.)

- O conjunto de palavras, compostos de acordo com essas regras e símbolos, constituem o **código fonte** de um software. Esse código fonte é traduzido para **código de máquina**, que é executado pelo microprocessador.



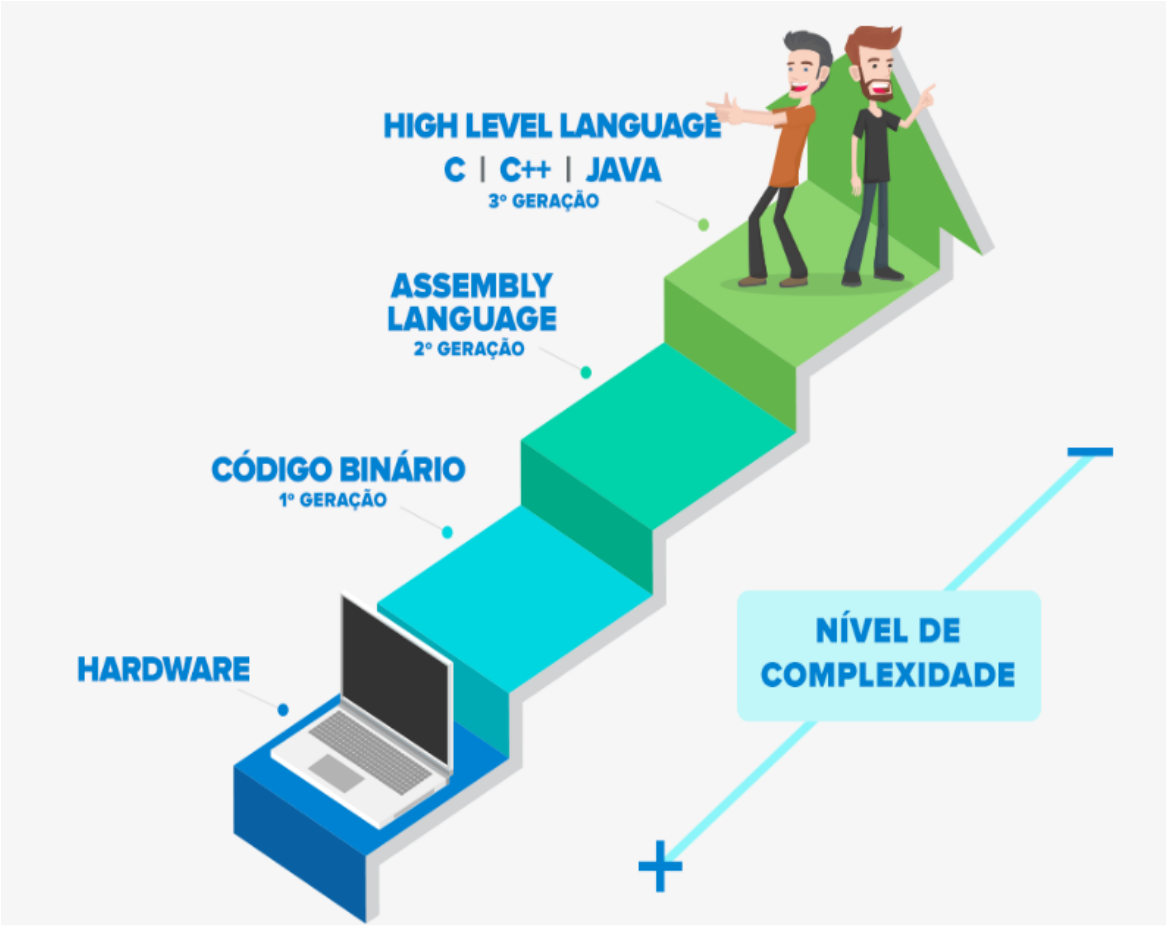
5. Linguagens de programação (cont.)

- A primeira e mais primitiva linguagem de computador é a própria linguagem de máquina, aquela que o computador entende diretamente e pode ser diretamente executada pelos circuitos do processador (pelo *hardware*).



5. Linguagens de programação (cont.)

- **Tipos de linguagens de programação:**
  - Baixo nível
  - Alto nível



5. Linguagens de programação (cont.)

- **Tipos de linguagens de programação:**
  - **Baixo nível**
    - A linguagem de baixo nível está mais próxima da linguagem de máquina.
      - **Exemplo:** *Assembly Language.*
    - São importantes para a área de Segurança da Informação.
      - Análises de códigos maliciosos;
      - Pesquisas e desenvolvimento de novas tecnologias.

Pseudo-código	Assembly (Intel 8088)
`leia(num)`	`MOV CX,0`
`para n de 1`	`IN AX,PORTA`
`até 10 passo 1`	`MOV DX,AX`
`faça`	`LABEL:`
`tab←num*n`	`INC CX`
`imprima(tab)`	`MOV AX,DX`
`fim-para;`	`MUL CX`
	`OUT AX, PORTA`
	`CMP CX,10`
	`JNE LABEL`

#include <stdio.h> int main() { // printf() displays the string inside quotation printf("Hello, World!"); return 0; }

5. Linguagens de programação (cont.)

- **Tipos de linguagens de programação:**
  - **Alto nível**
    - Mais próximas da linguagem natural.
    - Idealizada para a resolução de problemas sem preocupação com o tipo de CPU, memória, etc.
      - **Exemplos:** C, C#, Python, Java, PHP, JavaScript.
    - Quando se escreve uma instrução em alto nível, muitas conversões são necessárias para alcançar a linguagem de máquina.

```c
#include <stdio.h>

int main()
{
    printf("Hello World!");

    return 0;
}
```

5. Linguagens de programação (cont.)

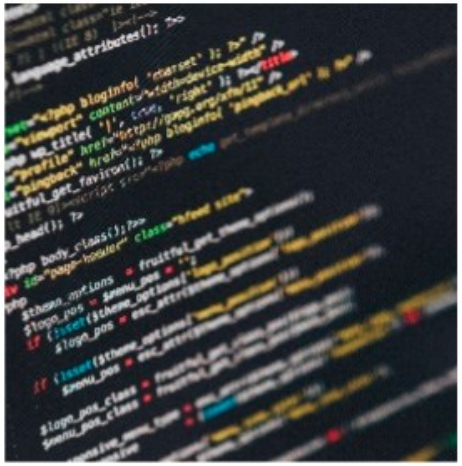
- Exemplos de código *"Hello World!"*
  - Em Python (alto nível):

```python
print("Hello World!")
```

  - Em Assembly (baixo nível):

```assembly
lea si, string
call printf
hlt
string db "Ola mundo!", 0
printf PROC
   mov AL, [SI]
   cmp AL, 0
   je pfend
   mov AH, 0Eh
   int 10h
   inc SI
   jmp printf
   pfend:
   ret
printf ENDP
```

5. Linguagens de programação (cont.)

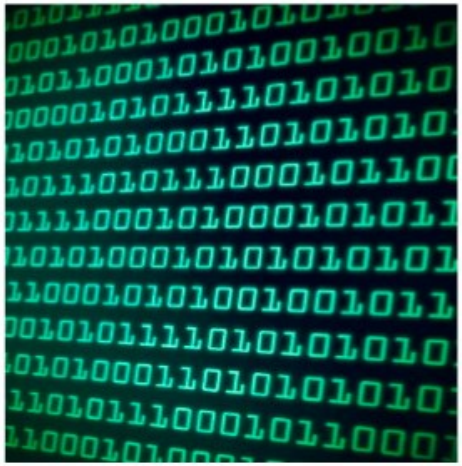


Alto Nível

- Aprendizado facilitado
- Menor custo de elaboração e manuntenção de software

Baixo Nível

- Aprendizado dificultado
- Melhor aproveitamento da arquitetura da máquina e mais velocidade de processamento



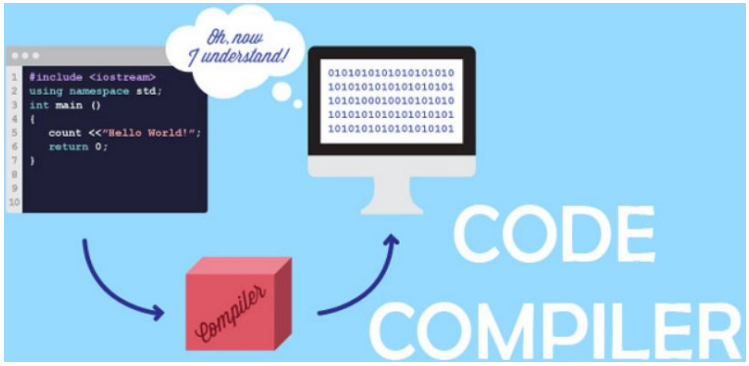
5.1. Tipos de código

- **Código-Fonte (ou programa-fonte)**
  - É o código do programa, na forma em que ele foi escrito.
  - É um arquivo texto contendo instruções em uma linguagem de programação.
  - Precisa ser convertido em linguagem de máquina para que possa ser executado pelo computador.

- **Código-Executável (ou programa-executável)**
  - É o código fonte do programa convertido para o formato binário/linguagem de máquina.

5.1. Tipos de código (cont.)

- Para produzir o código fonte de um programa é necessário ter um **editor de textos** e um **compilador** ou **interpretador**.
- Editor de textos pode ser **qualquer um que produza textos** não formatados.
- Há editores específicos (**IDE** - *Integrated Development Environment*) para uma ou várias linguagens. Eles **reconhecem a sintaxe dos comandos** dessas linguagens.
- Os **compiladores** ou **interpretadores** traduzem o código fonte de um programa para uma linguagem compreendida pelo computador.

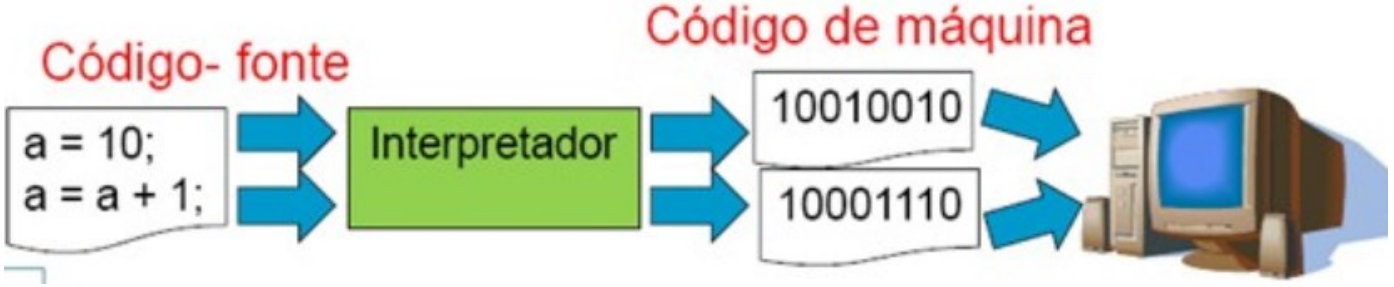


5.2. Do código fonte ao executável

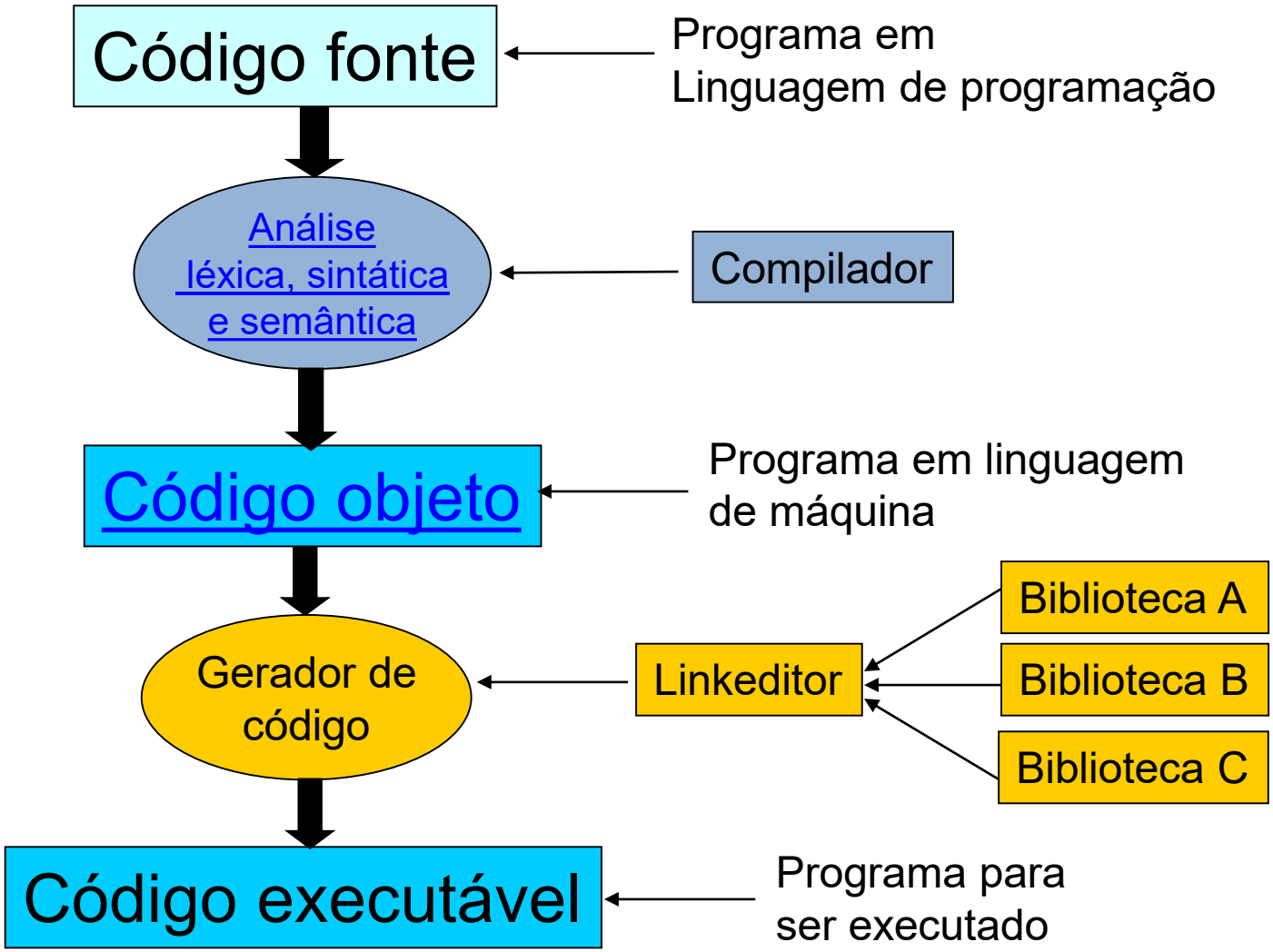
- De maneira geral, as linguagens de programação podem ser:
  - **Compiladas**
  - **Interpretadas**
- Esse processo permite transformar o código fonte em um programa que possa ser executado pelo computador.

Compiladas

Interpretadas

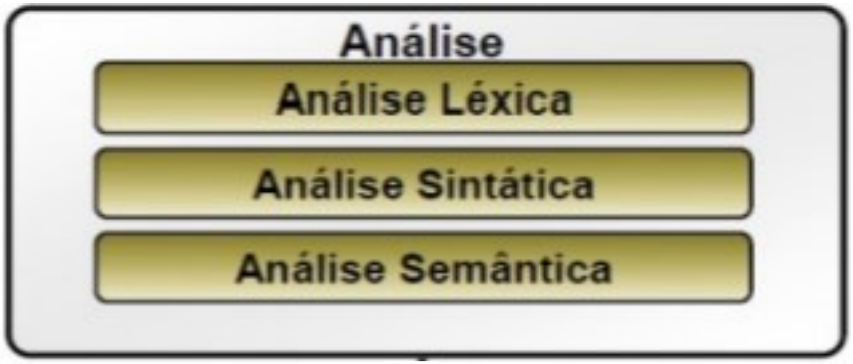


5.3. Processo de compilação



5.3. Processo de compilação (cont.)

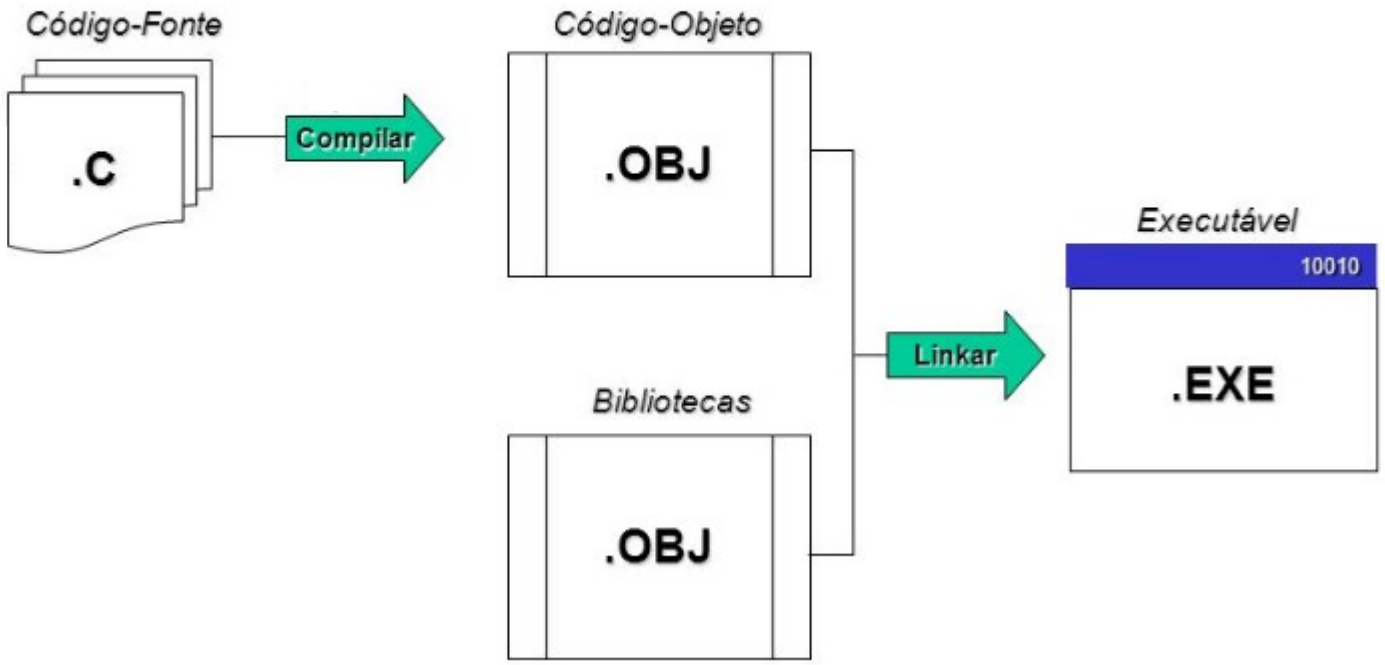
- As fases de análise (léxica, sintática, semântica) verificam:
  - Se o código-fonte é válido. Nesta fase, são acusados erros como:
    - ▶ Erros léxicos
      - Caracteres inválidos
        - "fi" ao invés de "if"
    - ▶ Erros sintáticos
      - Falta de ponto-e-vírgula
    - ▶ Erros semânticos
      - Variável não declarada
      - Incompatibilidade de tipos



  - Em caso de erros, o compilador não é capaz de "entender" o programa.
    - ▶ Tradução é impossível.

5.3. Processo de compilação (cont.)

- C é uma linguagem compilada



5.4. Linguagem compilada

- Na **linguagem compilada**, o compilador:
  - Lê a primeira instrução;
  - Realiza a análise do código;
  - Se não houver erro, transforma o código fonte em linguagem de máquina;
  - Repete o processo, instrução por instrução até o fim do código fonte ou encontrar um erro;
  - Gera um arquivo **objeto** com as instruções traduzidas;
  - Agrega outras rotinas traduzidas e gera um arquivo **executável**.

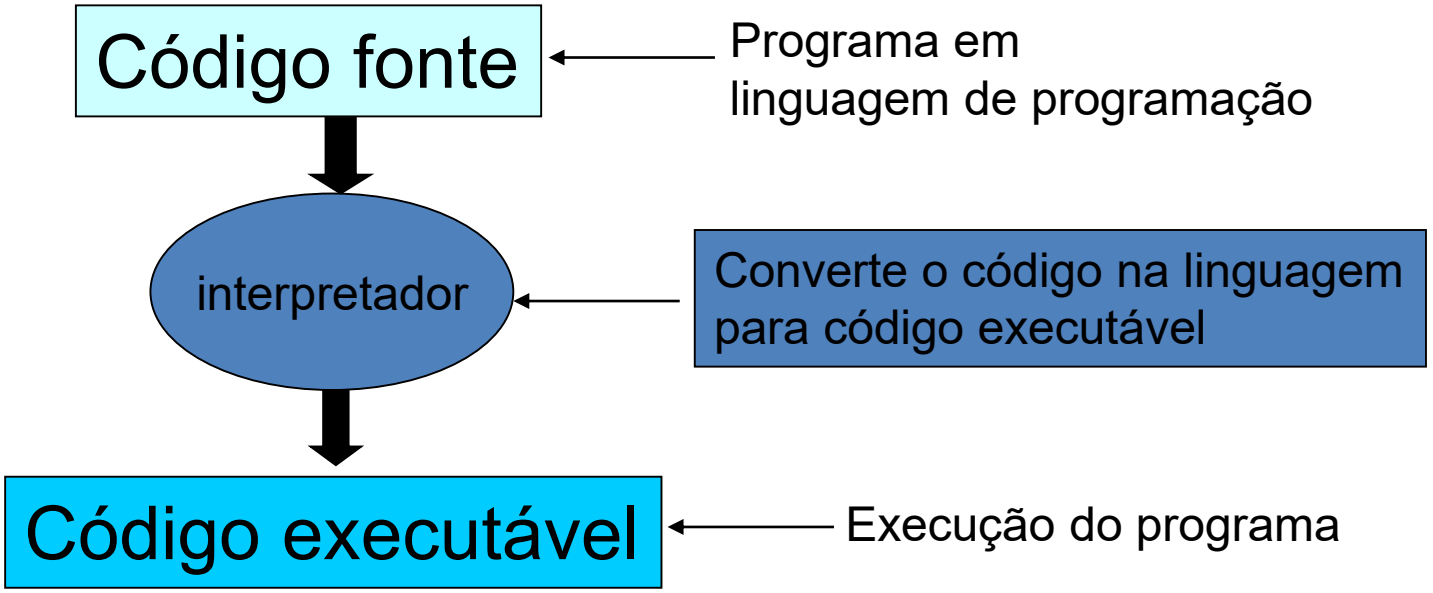
5.4.1. Processo de montagem

- O **processo de montagem** traduz o código fonte de um programa (linguagem de programação) para linguagem de máquina.
- Substitui os códigos de instruções simbólicas em linguagem de programação para valores numéricos.
- **Reserva espaço na memória** para realizar as instruções.
- Examina **sintaticamente** cada instrução do código fonte.

5.4.2. Processo de ligação ou linkedição

- A ligação ou linkedição **é útil para reusar funções** (partes) de outros programas já implementados.
  - **Exemplo:** funções de entrada e saída.
- O código é buscado onde estiver armazenado e é incorporado ou vinculado ao programa.
- Os códigos podem estar armazenados em bibliotecas.
- O processo de linkedição resulta em um conjunto de códigos de máquina interligados e prontos para funcionar.

5.5. Processo de Interpretação

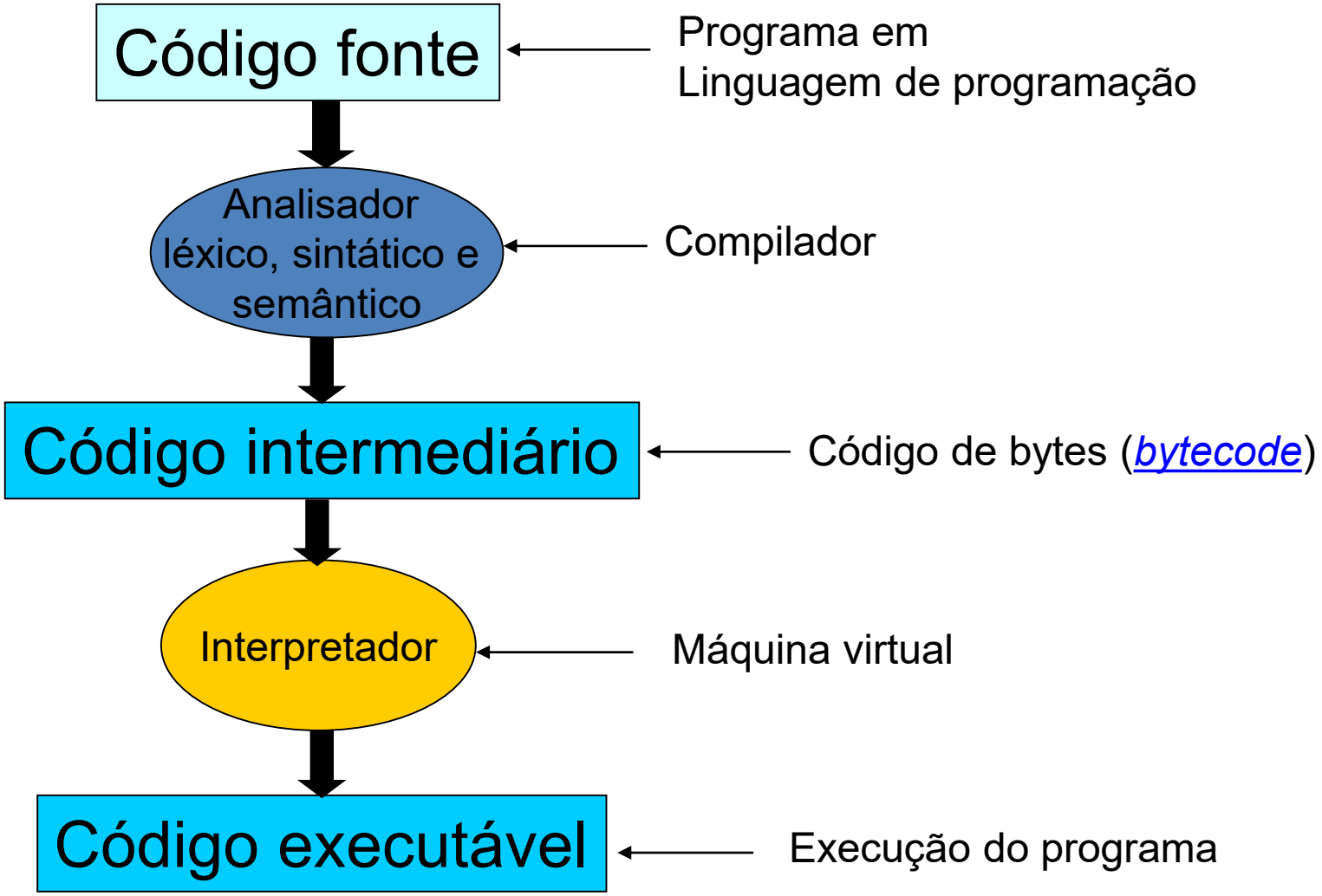


5.6. Linguagem interpretada

- Na **linguagem interpretada**, o interpretador:
  - Lê o código fonte instrução por instrução;
  - Realiza a análise sintática do código;
  - Transforma o código fonte em linguagem de máquina;
  - Executa a instrução;
  - Repete o processo até o fim do código fonte ou até encontrar um erro.

Uma linguagem interpretada: lê o código fonte, traduz e executa cada instrução cada vez que o programa é executado.

5.7. Processo híbrido

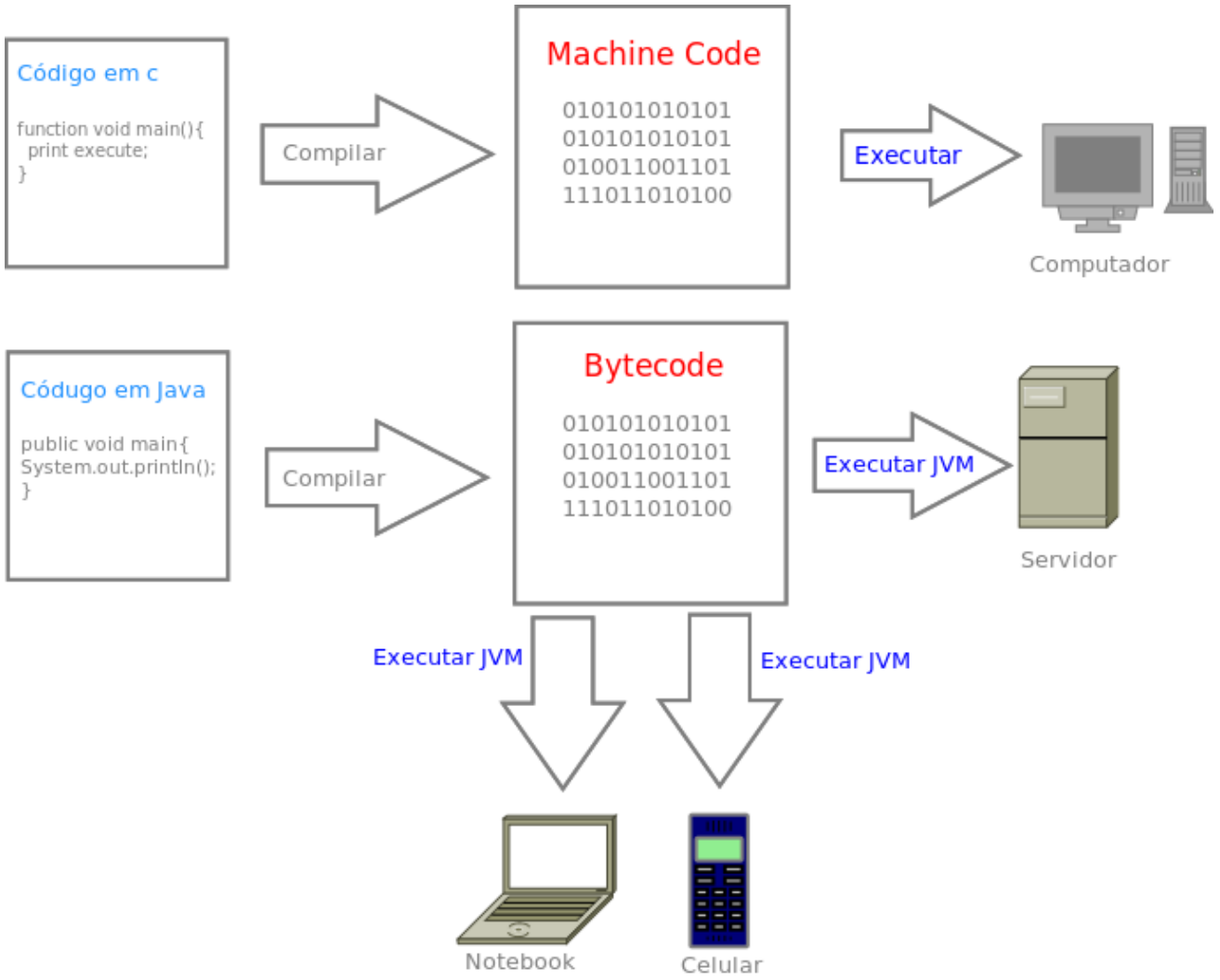


5.7 Linguagem compilada/interpretada

- Em **linguagens** que combinam **compilação** e **interpretação**:
  - Traduz as instruções para uma linguagem binária "universal" (***bytecode***).
  - O interpretador (**máquina virtual**) interpreta o código intermediário para que possa ser executado no sistema operacional.
    - A **máquina virtual** deve sempre estar presente para que a execução do programa ocorra e é dependente do sistema operacional.

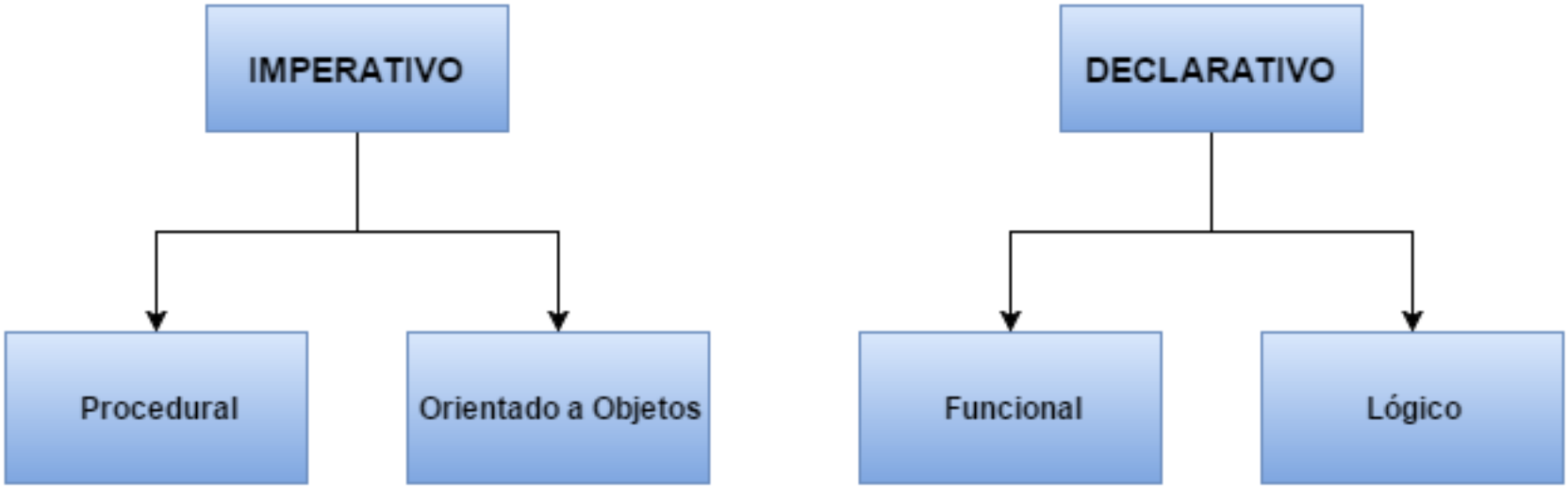
A ideia é criar programas que possam ser escritos uma vez e executados em qualquer plataforma, reduzindo os custos de desenvolvimento.
Apenas o software da máquina virtual é que precisa ser reescrito para cada plataforma, mas isso é de responsabilidade dos desenvolvedores da linguagem.

5.7 Linguagem compilada/interpretada (cont.)



5.8. Paradigmas de programação

- Um **paradigma de programação** está relacionado à forma de definir a solução de um problema.
- É um modo de classificar as linguagens de programação.



5.9. Paradigmas de programação (cont.)

- **Imperativo**
  - Ações ou comandos que mudam o estado (variáveis) de um programa.
  - Está ligado ao tempo verbal imperativo, onde o programador diz ao computador: ***faça isso***, ***depois isso***, ***depois aquilo***...
    - **Procedural:** C, Pascal
    - **Orientado a objetos:** C++, Java
- **Declarativo** – Permite o desenvolvedor definir o que o programa ***deve*** realizar ao invés de definir exatamente ***como*** ele deve realizá-lo.
    - **Funcional:** Fortran, LISP
    - **Lógico:** Prolog
      - **Exemplo em Prolog** (ênfase em regras e fatos)**:**

        avo(X,Z) :- pai(X,Y), pai(Y,Z).

        (X é avô de Z se X é pai de Y e Y é pai de Z)

5.9 Programação procedural (cont.)

- O **paradigma imperativo** preconiza que todos os programas possíveis podem ser reduzidos a apenas três estruturas:
  - Sequência
  - Decisão
  - Repetição

- A **modularização** procura segmentar **um problema complexo** em partes menores e, portanto, mais simples.

Referências

- KERNIGHAN, B. W., RITCHIE, D. M. **C: A linguagem de programação**. Rio de Janeiro: Campus, 1986.
- SEBESTA, R. W. **Conceitos de linguagens de programação**, 11ª ed. Porto Alegre: Bookman, 2003.
- Conteúdo baseado no material disponibilizado pela professora Beatriz Borsoi.

Dúvidas

- ???