UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS DE RUSSAS

RUS0059 - Linguagens de Programação

Implementando uma linguagem

Profa. Elanne Mendes

Conteúdo da aula

- Conceitos
- Compilação
- Interpretação Pura
- Interpretação Híbrida

Sintaxe

- Define a forma de um programa estruturalmente correto.

Sintaxe

- Define a forma de um programa estruturalmente correto.
  - Qual o conjunto básico de **palavras** e **símbolos** que os programadores usam para escrever programas nessa linguagem?
  - Como essas palavras e símbolos podem ser **associados**?
  - De que forma eles devem ser **organizados**?

Sintaxe

- Define a forma de um programa estruturalmente correto.
  - Qual o conjunto básico de **palavras** e **símbolos** que os programadores usam para escrever programas nessa linguagem?
  - Como essas palavras e símbolos podem ser **associados**?
  - De que forma eles devem ser **organizados**?

```
for(<inicializacao> ; <condicao> ; <passo>) <comando>
```

Sintaxe

- Define a forma de um programa estruturalmente correto.

C

```c
#include <stdio.h>
int main(void)
{
    printf("Hello, world!\n");
    return 0;
}
```

Lua

```lua
print("Hello, world!")
```

COBOL

```
000100 IDENTIFICATION DIVISION.
000200 PROGRAM-ID.    HELLOWORLD.
000900 PROCEDURE DIVISION.
001000 MAIN.
001100   DISPLAY "Hello, world!".
001200   STOP RUN.
```

Fortran

```
program hello
    write(*,*) "Hello, world!"
end program hello
```

Java

```java
class Hallo {
  public static void main( String[] args ) {
    System.out.println("Hello, world!");
  }
}
```

Nomes

- O vocabulário de uma linguagem inclui um conjunto de **regras** para nomear entidades.
  - Variáveis
  - Funções
  - Parâmetros
  - Tipos estruturado
  - Classes
  - ...

Nomes

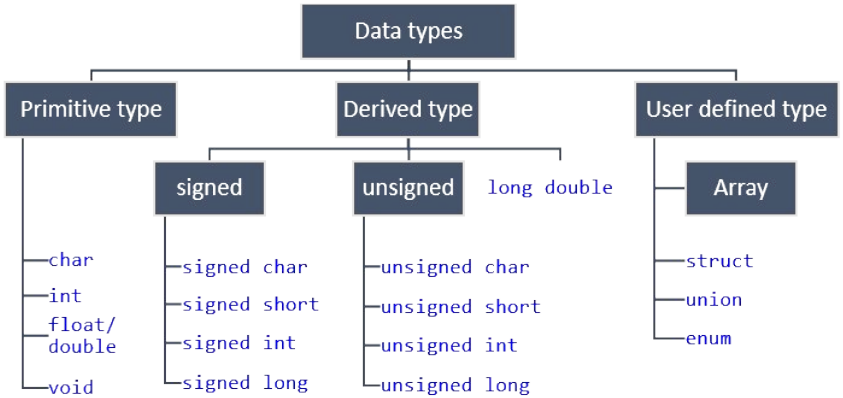
- O vocabulário de uma linguagem inclui um conjunto de **regras** para nomear entidades
  - Variáveis
  - Funções
  - Parâmetros
  - Tipos estruturado
  - Classes
  - ...

Na linguagem C

- *Os caracteres permitidos são: números, letras e o caractere sublinhado*
- *O primeiro caractere deve ser uma letra ou sublinhado*
- *Não são permitidos espaços em branco e caracteres especiais (@, $, +, %, ...)*
- *Há diferença entre letras minúsculas e maiúsculas (Case Sensitive)*

Tipos

- Tipos de uma linguagem denotam os tipos de valores que os programas podem **armazenar** e **manipular**.
  - Simples → Inteiros, Reais, Caracteres, Lógicos
  - Estruturados → Strings, Vetores, Listas, Árvores, ...
  - Complexos → Classes

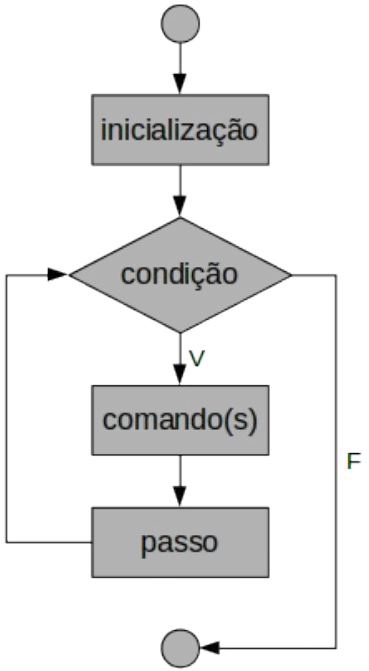


Semântica

- Define o significado das estruturas de uma linguagem.
  - Qual o efeito de cada comando sobre os valores das variáveis?
  - Qual o comportamento das funções?
  - Como a memória é gerenciada?

Semântica

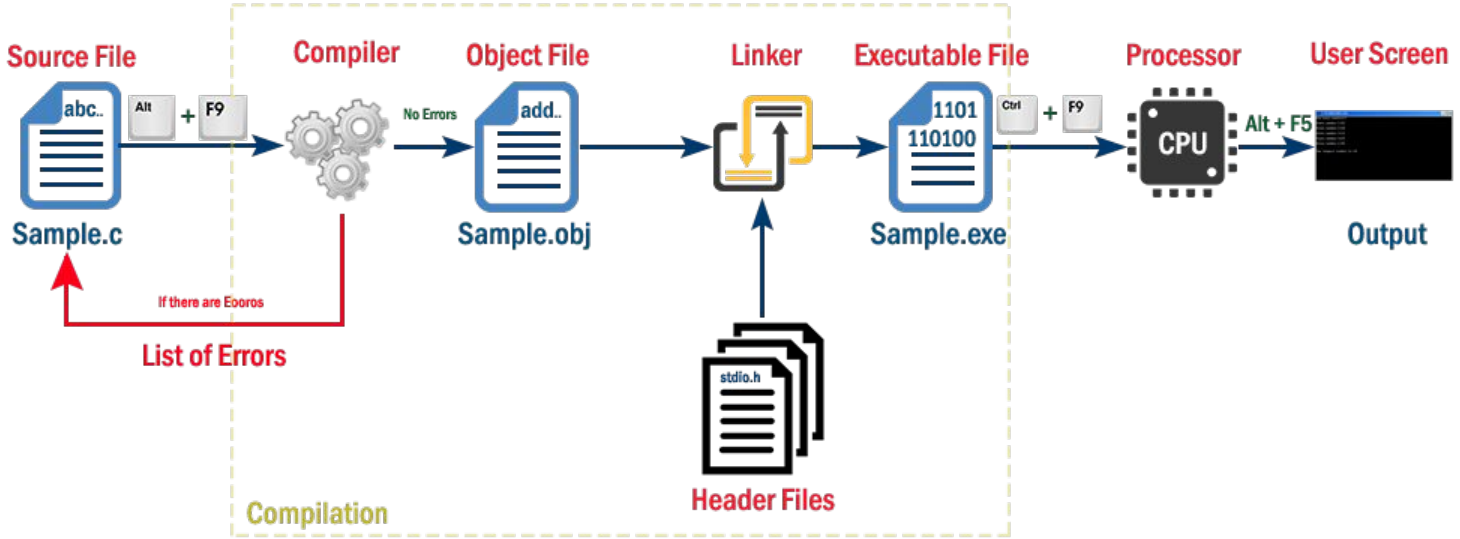
- Define o significado das estruturas de uma linguagem.
  - Qual o efeito de cada comando sobre os valores das variáveis?
  - Qual o comportamento das funções?
  - Como a memória é gerenciada?




```
for(<inicializacao> ; <condicao> ; <passo>) <comando>
```

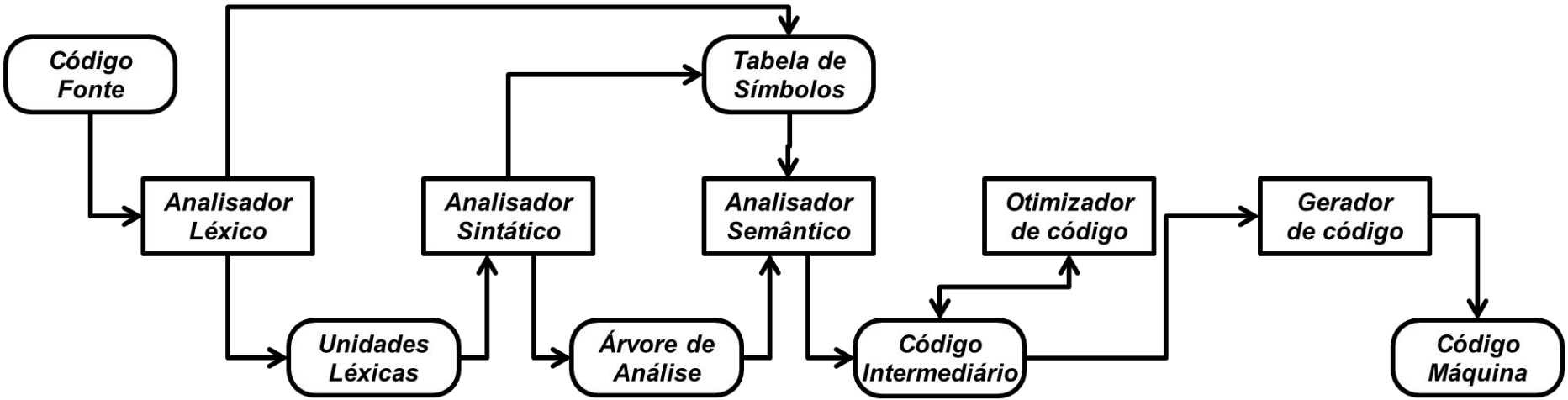
Compilação

- Processo de tradução de um **programa-fonte** escrito em uma linguagem de **alto nível** para linguagem de **máquina.**



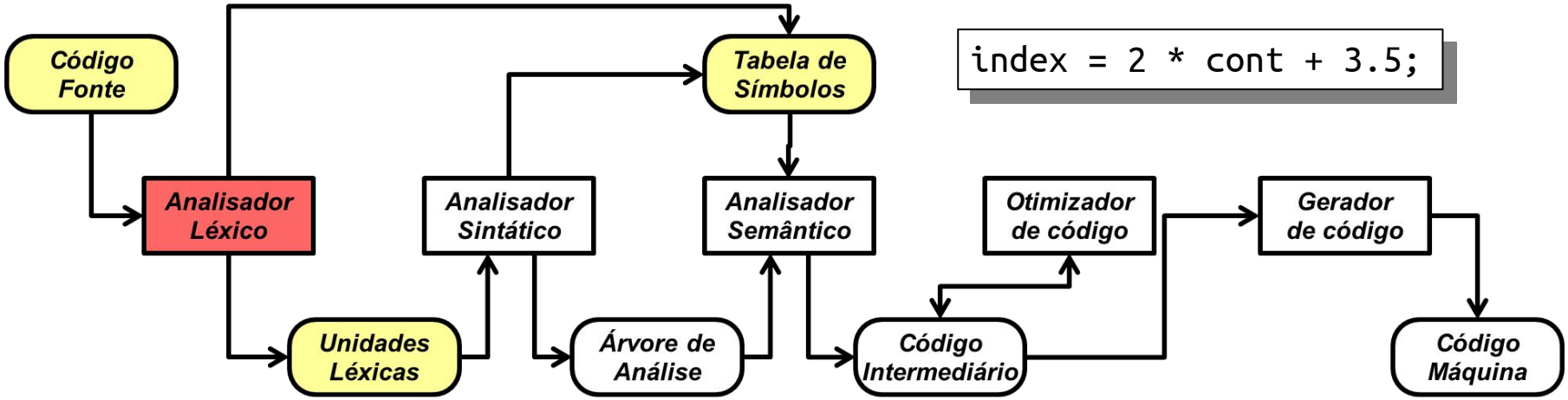
Compilação

- Processo de tradução de um **programa-fonte** escrito em uma linguagem de **alto nível** para linguagem de **máquina.**



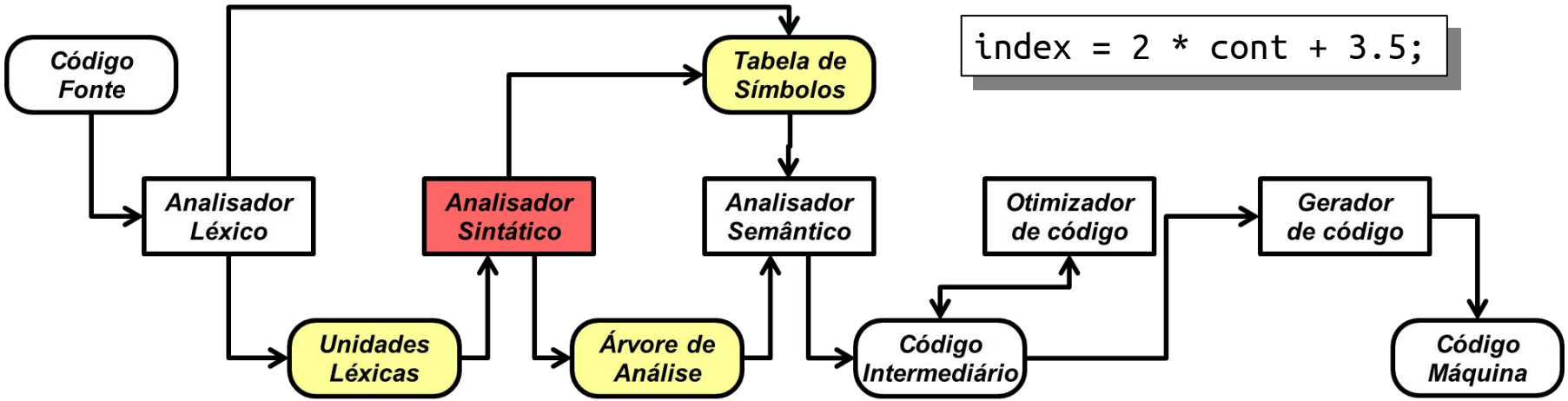
Compilação

- A análise léxica é a primeira etapa da compilação, na qual o **código fonte** é analisado e dividido em "*tokens*" (unidades léxicas) que representam **palavras-chave**, **identificadores**, **números**, **símbolos de pontuação**, etc.



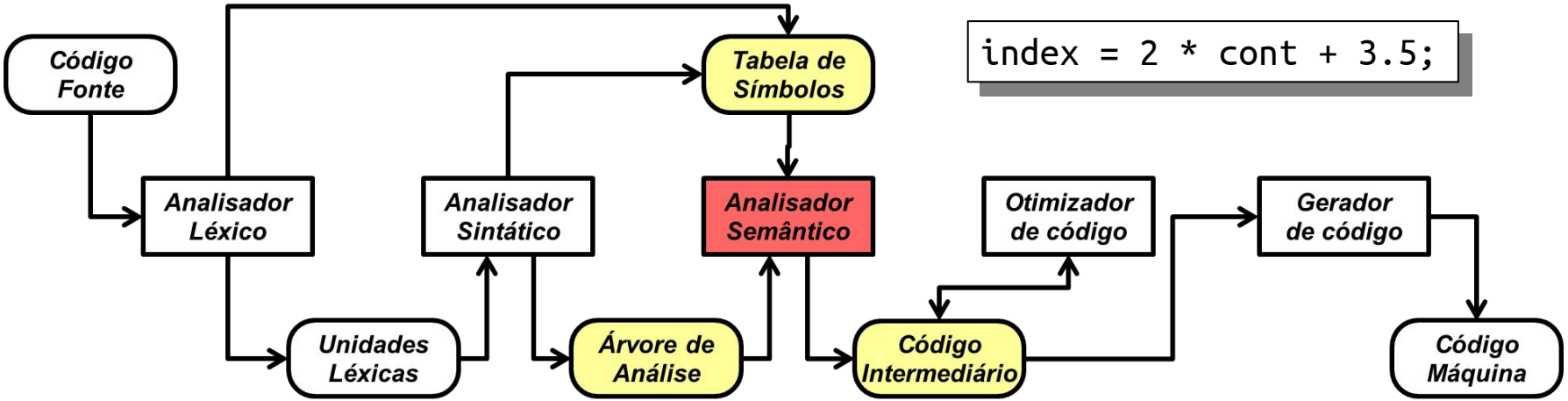
Compilação

- A análise sintática é a segunda etapa da compilação, na qual a estrutura do programa é verificada em relação à **gramática** da linguagem de programação. Responsável por identificar ***erros de sintaxe mais complexos***, como uso **incorreto de operadores**, **problemas de precedência** (ou seja, ordem das instruções), etc.



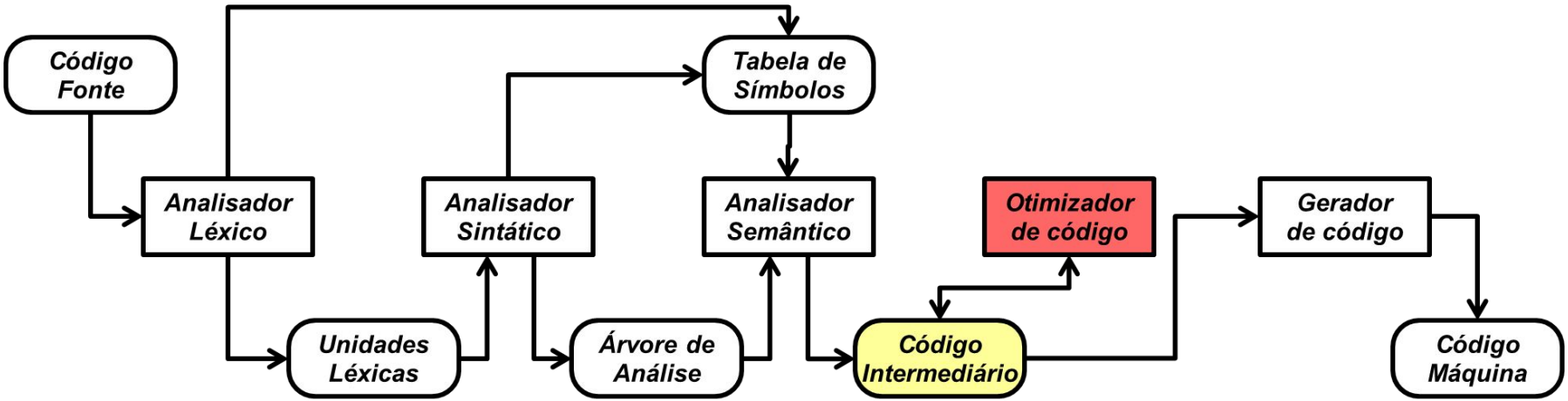
Compilação

- A análise semântica é a terceira etapa da compilação, na qual a estrutura do programa é verificada em relação às **regras semânticas** da linguagem de programação. Essa análise envolve a verificação de ***tipos de dados***, escopo de **variáveis**, checagem de **compatibilidade entre os tipos de dados**, etc.



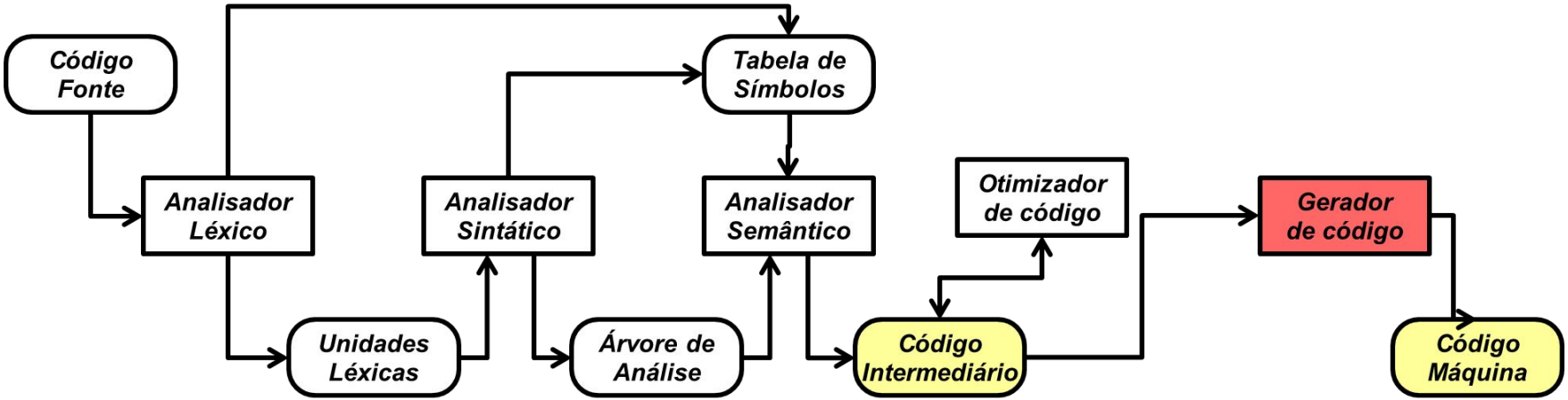
Compilação

- Processo de tradução de um **programa-fonte** escrito em uma linguagem de **alto nível** para linguagem de **máquina.**



Compilação

- Processo de tradução de um **programa-fonte** escrito em uma linguagem de **alto nível** para linguagem de **máquina.**



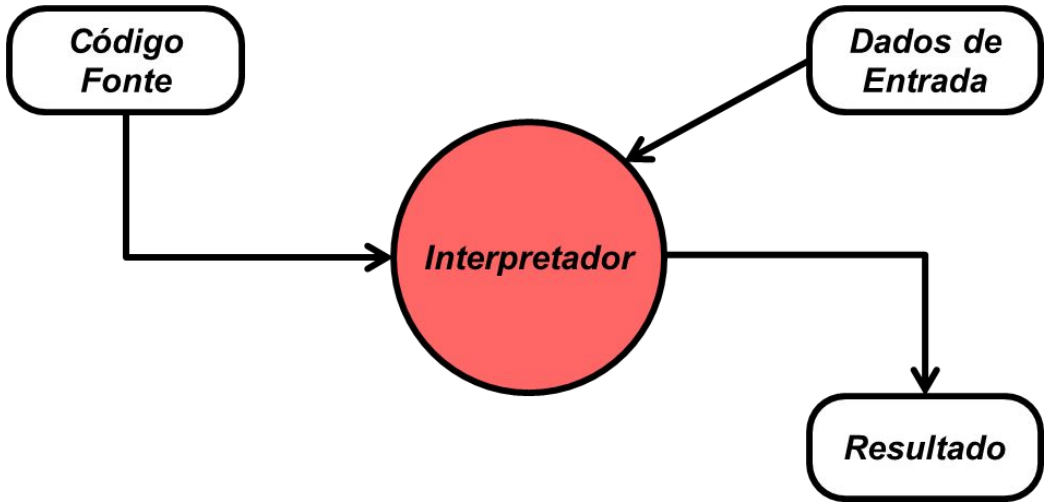
Compilação

- Processo de tradução de um **programa-fonte** escrito em uma linguagem de **alto nível** para linguagem de **máquina**.

- **Vantagens:**
  - Execução mais rápida.
  - Pouco consumo de memória.
- **Desvantagens:**
  -
  - Dependência da plataforma do código de máquina gerado.

Interpretação

- Interpretador age como uma **máquina virtual**, tratando sentenças de alto-nível em vez de instruções de máquina.



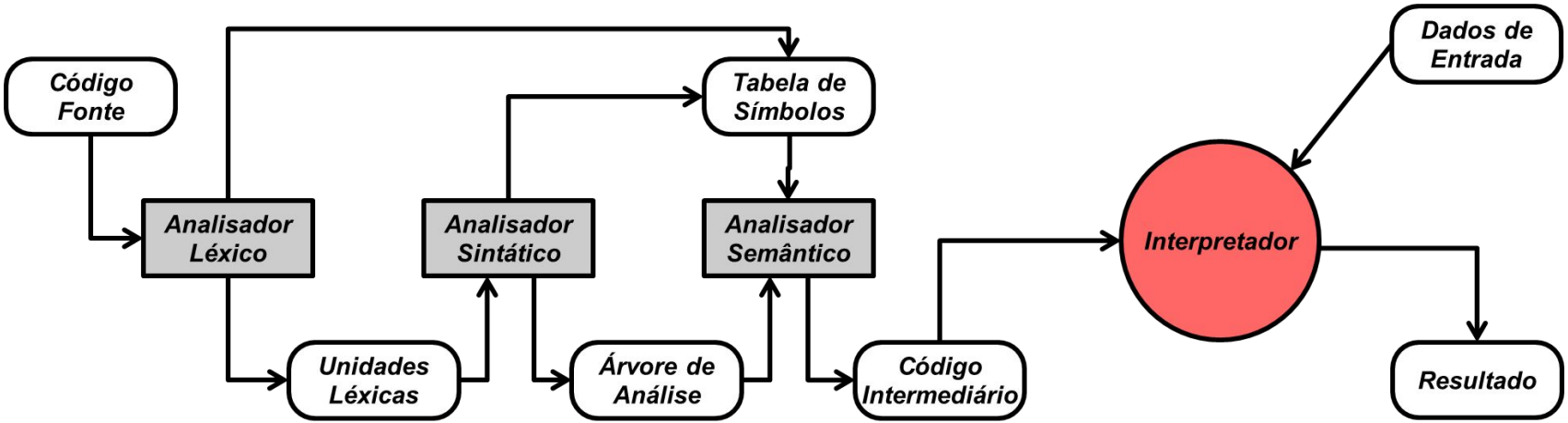
Interpretação

- Interpretador age como uma **máquina virtual**, tratando sentenças de alto-nível em vez de instruções de máquina.

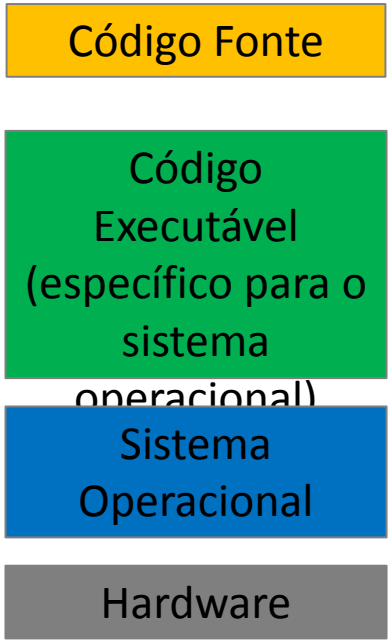
- **Vantagens:**
  - Editar o código durante a execução.
  - Erros podem ser identificados mais fácilmente.
- **Desvantagens:**
  - Execução mais lenta.
  - Consumo de memória maior.

Interpretação “Híbrida”

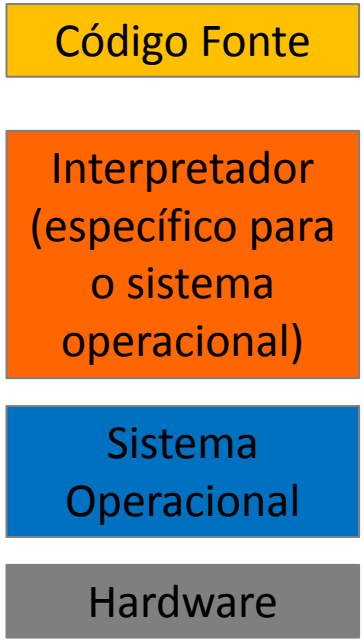
- Programa é traduzido para forma **abstrata intermediária**, que é então executada de modo interpretado.



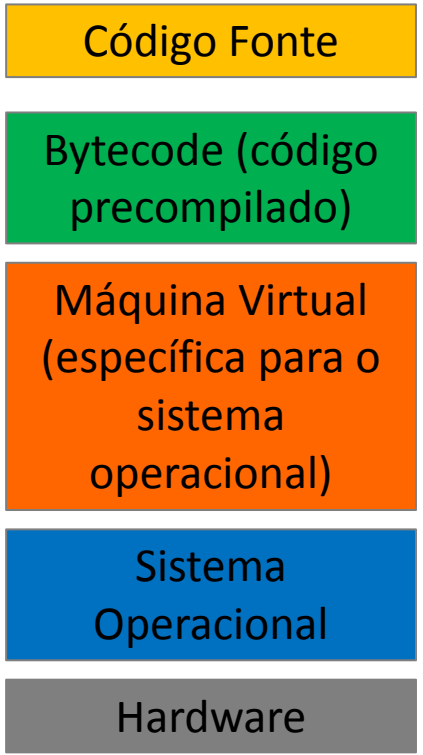
Compilação



Interpretação



Híbrido



Compilação vs Interpretação

Eficiência de Execução:

- **Compiladas**: Linguagens compiladas geralmente tendem a ser ***mais eficientes*** em termos de tempo de **execução**, pois o código já foi traduzido para **instruções de máquina** otimizadas durante a compilação. Isso resulta em **execução mais rápida**.
- **Interpretadas**: Linguagens interpretadas geralmente têm uma camada adicional de interpretação durante a execução, o que pode levar a um ***desempenho mais lento*** em comparação com as linguagens compiladas.

Compilação vs Interpretação

Portabilidade:

- **Compiladas:** Para cada plataforma (como diferentes sistemas operacionais ou arquiteturas de processadores), é necessário ***recompilar*** o código-fonte para gerar um ***novo executável compatível***. Isso pode ser menos portátil em comparação com linguagens interpretadas.

- **Interpretadas:** Geralmente, linguagens interpretadas são mais **portáteis**, pois um interpretador pode ser escrito para **cada plataforma** e o código-fonte é ***executado diretamente***, sem necessidade de recompilação.

Compilação vs Interpretação

Depuração e Desenvolvimento:

- **Compiladas:** Processo de compilação é **separado** do processo de execução. Isso pode fazer com que seja necessário um ciclo de compilação antes de ver as alterações refletidas no programa.
- **Interpretadas:** Linguagens interpretadas muitas vezes oferecem um ambiente de desenvolvimento **mais ágil**, permitindo que os desenvolvedores vejam imediatamente os **resultados das mudanças no código**.

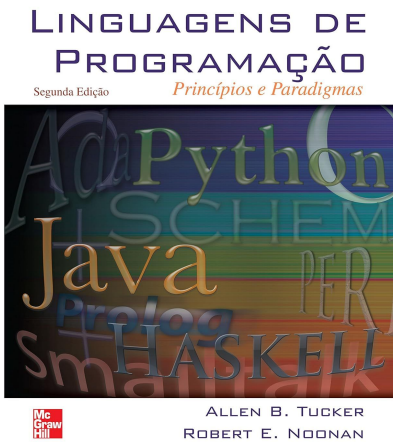
Exercício

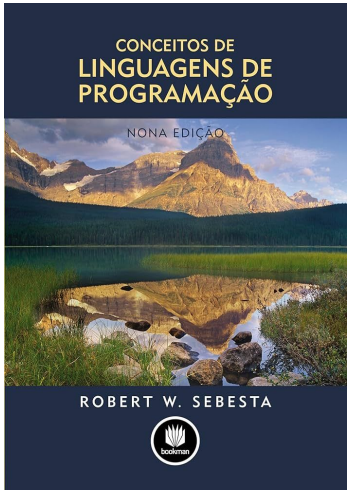
- Explique os conceitos de **Compilação** e **Interpretação**.
- Cite e descreva as vantagens e desvantagens da **Compilação** e **Interpretação**.

Resumo – Capítulo 1

- ❑ Princípios - História
- ❑ Paradigma
- ❑ Projeto de linguagem – características específicas
- ❑ Desafios de programação que envolvem a utilização efetiva de uma linguagem
- ❑ Implementação

Bibliografia recomendada





- **TUCKER, Allen B.; NOONAN, Robert. Linguagens de programação: princípios e paradigmas. 2. ed. São Paulo: McGraw-Hill, 2008.**
- **SEBESTA, Robert W. Conceitos de linguagens de programação. 9. ed. Porto Alegre: Bookman, 2011.**

Próxima aula...

- **Capítulo 2 – Sintaxe**
  - **2.1 Gramáticas**

Obrigada!

Profa. Elanne Mendes
elanne@ufc.br

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS DE RUSSAS