Marisangila Alves, MSc
marisangila.alves@edu.sc.senai.br
marisangila.com.br

2026/1

Lógica de Programação

Liguagens de Programação

Sumário

Programa

Computador:

É um dispositivo capaz de realizar cálculos e tomar decisões lógicas com uma velocidade milhões ou mesmo bilhões de vezes mais rápida do que os seres humanos (Deitel; Deitel, 2011).

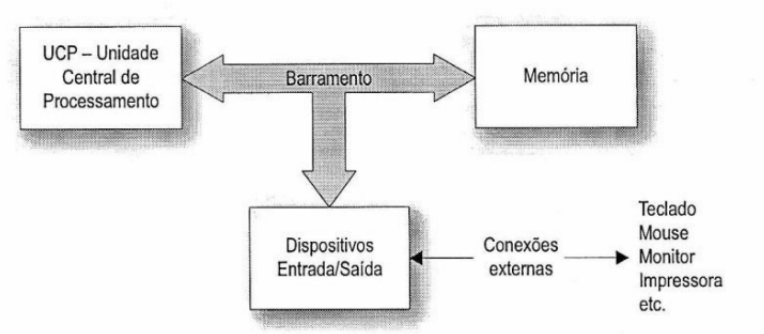


Figura 1: Arquitetura simplificada de um computador (Medina; Fertig, 2005).

Recapitulando...

> *Os computadores processam dados sob o controle de conjuntos de instruções chamados programas de computador. [...] Os vários dispositivos (como teclado, tela, discos, memória e unidades de processamento) que constituem um sistema computacional são chamados de hardware. Os programas executados em um computador são chamados de software.*
>
> (Deitel; Deitel, 2011)

Função básica de um computador (De Oliveira; Manzano, 2004).

- Entrada: Coleta de dados do mundo real, geralmente por meio de entrada manual.
- Processamento: Realização de operações e transformações nos dados.
- Saída: Apresentação visual ou armazenamento dos resultados no mundo real.

> *Todo programa de computador deve, de alguma forma, possibilitar a entrada dos dados do mundo exterior, produz ir a ação de processamento, tanto lógico quanto matemático, sempre que essas ações forem necessárias, e acima de tudo possibilitar a saída de dados que tenham sido processados ou estej am apenas armazenados.*
>
> (De Oliveira; Manzano, 2004)

O que é um Programa?

> *Um conjunto de instruções que será executado pelo processador em uma determinada sequência. Esse programa leva o computador a executar alguma tarefa.*
>
> (Medina; Fertig, 2005)

Resumindo:

Para que um algoritmo escrito em pseudocódigo ou representado como um fluxograma seja executado, é necessário que ele seja traduzido para uma linguagem de programação. Uma sequência de instruções escritas em uma linguagem de programação e armazenada em um arquivo constitui um programa, que, quando executado, se torna um *software*.

Linguagem de Programação

O que é uma Linguagem de Programação?

- A linguagem de programação que um computador é capaz de compreender é composta apenas de números.
- Sendo assim, pode ser extremamente complicado, para nós, seres humanos, criarmos algoritmos na linguagem de máquina.
- Por esse motivo, existem linguagens de programação que podem ser compreendidas por seres humanos.
- No entanto, para que o programa possa ser executado, a linguagem de programação deve ser traduzida para linguagem de máquina.
- As linguagens de programação têm uma sintaxe específica, e são por definição, no idoma Inglês[1].

[1] Linguagens de programação são sempre escritas no idioma inglês, pelo fato de a ideia de constituição de hardware e de software ter sido proposta por cientistas ingleses (Charles Babbage e Ada Augusta)

Ferramentas de Programação

- **Editor de Texto:**
  - É uma ferramenta simples usada para implementar[2] e editar código-fonte;
  - Genérico, pode ser configurado para qualisquer linguages de programação.
- **IDE (Ambiente de Desenvolvimento Integrado):**
  - É uma ferramenta completa que combina editor de texto, compilador ou interpretador, depurador e outras funcionalidades para facilitar o desenvolvimento de *software*.
  - Normalmente, é específico para apenas uma linguagem de programação.

[2] **Implementar** é o processo de escrever e desenvolver o código necessário para um programa.

(a) Atom (GitHub, 2024)

(b) VIM (Vim, 2024)

(c) Visual Studio Code (Microsoft, 2024)

(d) Sublime (Sublime, 2024)

Figura 2: Editores de texto.

(a) Android Studio (Google, 2025)

(b) Eclipse (Foundation, 2025)

(c) IntelliJ IDEA (JetBrains, 2025a)

(d) NetBeans (Medina; Fertig, 2005)

(e) PyCharm (JetBrains, 2025b)

(f) Visual Studio (Microsoft, 2025)

(g) Xcode (Apple, 2025)

(h) R Studio (RStudio, 2025)

Figura 3: Principais IDEs para desenvolvimento de *software*.

Alto e Baixo nível

- **Linguagem alto nível:**
  - São mais próximas da linguagem humana ou escrita convencional.
  - Exemplos: Python, Java, C, C++, C#, PHP, R, Go.
  - São utilizadas na maioria dos *softwares*: aplicativos, websites, sistemas operacionais e dispositivos eletrônicos.
- **Linguagem baixo nível:**
  - Mais próximas da linguagem de máquina, com instruções específicas para o *hardware*.
  - Exemplo: Assembly.
  - Utilizado em programas que interagem diretamente com o *hardware* do computador ou outros dispositivos.
- **Linguagem de máquina:**
  - É a linguagem nativa dos computadores, composta exclusivamente por números binários (0s e 1s).
  - Executada diretamente pela Unidade Central de Processamento (CPU) sem necessidade de tradução.
  - Extremamente difícil de ler e escrever diretamente por humanos.

Linguagem de máquina	Linguagem de baixo nível	Linguagem de alto nível
0010 0001 1110	LOAD R1, a	`a = a + b`
0010 0010 1111	LOAD R2, b	
0001 0001 0010	ADD R1, R2	
0011 0001 1111	STORE R1, a	

Tabela 1: Comparação entre linguagens de máquina, baixo nível e alto nível.

Alto e Baixo nível III

```python
a = 5
b = 3
a = a + b
```

Python.

```assembly
section .data
    a dq 5
    b dq 3
section .text
    global _start
_start:
    ; LOAD R1, a
    mov rax, [a]
    ; LOAD R2, b
    mov rbx, [b]
    ; ADD R1, R2
    add rax, rbx
    ; STORE R1, a
    mov [a], rax
    mov rax, 60
    xor rdi, rdi
    syscall
```

Assembly.

Baixo nível

Linguagem Assembly

A linguagem **Assembly** é uma linguagem de programação de **baixo nível**, muito próxima da linguagem de máquina. Cada instrução corresponde diretamente a uma operação executada pelo processador.

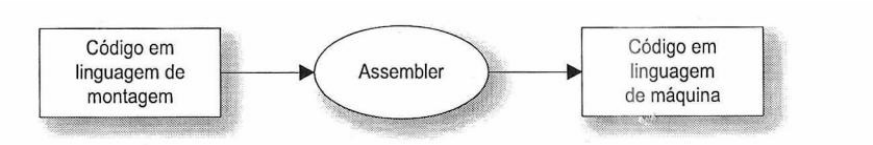


Figura 4: Montagem para linguagem de máquina citemedina2005algoritmos.

Características

- instruções específicas da arquitetura do processador;
- manipulação direta de registradores e memória;
- grande controle sobre o hardware;
- código traduzido para binário por um **montador (assembler)**.

Exemplos:

- desenvolvimento de sistemas operacionais;
- drivers de hardware;
- sistemas embarcados;
- otimizações críticas de desempenho.

O conjunto de instruções disponíveis mudam de acordo com a arquitetura do hardware.

```
section .text
global _start

_start:
    mov rax, 5          ; rax = 5
    imul rax, rax       ; rax = rax * rax (25)

    mov rax, 60         ; syscall exit
    xor rdi, rdi
    syscall
```

Código 1: Exemplo em linguagem Assembly para x86.

```
.global _start

_start:
    mov x0, #5          // x0 = 5
    mul x0, x0, x0      // x0 = x0 * x0 (25)

    mov x8, #93         // syscall exit
    mov x0, #0
    svc #0
```

Código 2: Exemplo em linguagem Assembly para ARM.

Alto nível

Tradução

Compilação

Compilar é o processo de traduzir um programa escrito em uma linguagem de programação (código fonte) para linguagem de máquina antes da execução. Esse processo é realizado por um **compilador**, que gera um arquivo executável que pode ser executado diretamente pelo computador.

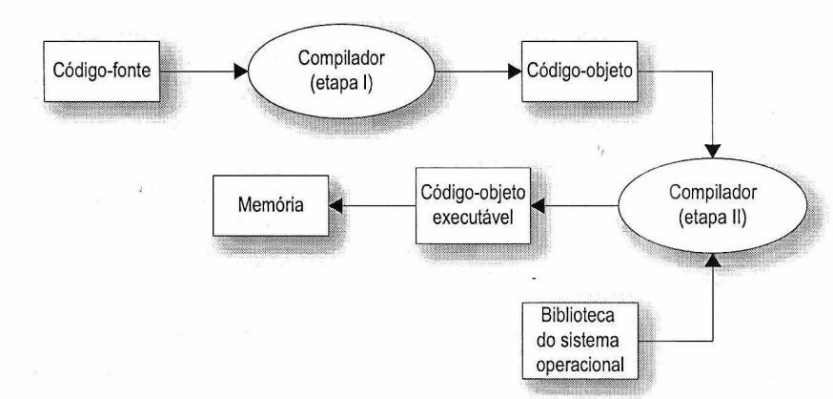


Figura 5: Execução de um programa em linguagem de alto nível (Medina; Fertig, 2005).

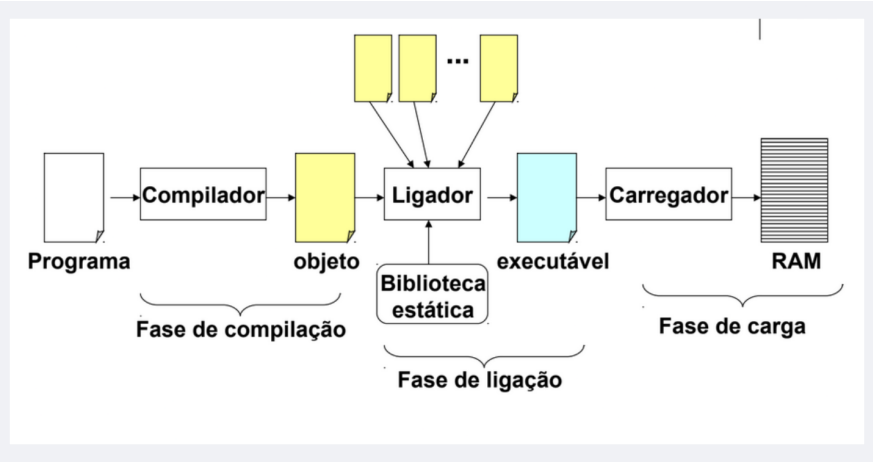


Figura 6: Processo de tradução e execução de um programa.

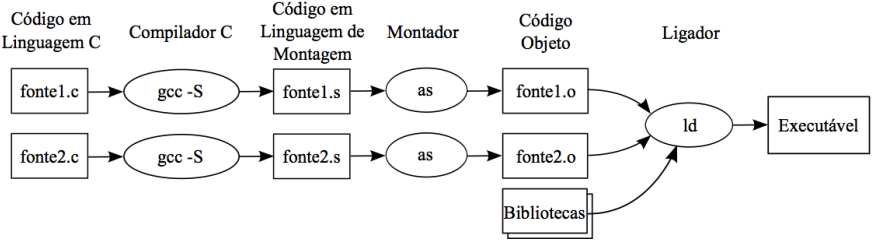


Figura 7: Processo de tradução e execução de um programa em C.

- **Montador**: programa que traduz a **linguagem de montagem** para **código de máquina**.
- **Linguagem de montagem**: representação simbólica da linguagem de máquina de um processador específico, acrescida de instruções adicionais que facilitam a escrita do programa e fornecem diretivas para o montador.
- **Compilador**: programa que converte um código escrito em uma **linguagem de programação** (linguagem fonte) para **linguagem de máquina** (código objeto). Pode oferecer recursos como variáveis automáticas, expressões aritméticas, estruturas de controle (`FOR`, `WHILE`), escopo de variáveis e funções de alto nível.

- **Código executável**: código de máquina gerado por um compilador ou montador, pronto para ser executado pelo computador.
- **Conjunto de instruções**: conjunto de todas as instruções de linguagem de máquina que um determinado processador é capaz de executar.
- **Ligador (Linker)**: programa utilitário que combina um ou mais arquivos de **código objeto** gerados separadamente em um único arquivo executável.

- **Carregador (Loader)**: rotina que carrega um programa executável na memória para que ele possa ser executado.
- **Linguagem de máquina (código de máquina)**: representação binária de um programa que é diretamente interpretada e executada pelo processador.
- **Código objeto**: versão em linguagem de máquina do código fonte gerada por um compilador ou montador, que posteriormente é transformada em código executável pelo ligador.

Interpretação

Interpretar é executar um programa traduzindo suas instruções passo a passo durante a execução. Nesse caso, um **interpretador** lê o código fonte e executa cada instrução diretamente, sem gerar um arquivo executável previamente.

Compilador traduz todo o programa antes da execução; interpretador traduz e executa o programa passo a passo durante a execução.

Característica	Compilador	Interpretador
Processamento	Traduz o programa inteiro antes de executar	Traduz e executa ao mesmo tempo
Arquivo executável	Gera executável	Não gera executável
Ferramentas auxiliares	Usa linker e loader	Não usa linker
Desempenho	Execução mais rápida	Execução mais flexível

Tabela 2: Comparação entre compiladores e interpretadores

Paradigmas de Programação

Paradigma de Programação

Um paradigma de programação é uma **forma de escrever programas**, que define como os problemas são estruturados e como as soluções são expressas em uma linguagem de programação.

- Diferentes paradigmas oferecem diferentes formas de pensar a solução de um problema.
- Muitas linguagens modernas suportam **mais de um paradigma**.

Exemplos:

- Imperativo ou Estruturado;
- Funcional;
- Orientado a Objetos;
- Orientado a Eventos.

A seguir linguagens de alto nível:

Imperativo

No paradigma imperativo o programa é escrito como uma **sequência de instruções** que modificam o estado do sistema.

- Uso de variáveis;
- Estruturas de controle (`if`, `while`, `for`);
- Execução sequencial.

Exemplos:

- C;
- Pascal;
- Fortran;
- Ada.

Funcional

Baseado no conceito de **funções matemáticas**. O programa é composto por funções que recebem dados e retornam resultados.

- Evita alteração de estado;
- Uso de funções puras;
- Recursão em vez de laços.

Exemplos:

- Haskell;
- Lisp;
- Scala.

Orientado a Objetos

Organiza o programa em **objetos** que representam entidades do mundo real, contendo atributos e comportamentos.

- Classes;
- Objetos;
- Encapsulamento;
- Herança;
- Polimorfismo.

Exemplos:

- Java;
- C++;
- Python;
- C#.

Orientado a Eventos

O fluxo do programa é controlado por **eventos**, como ações do usuário ou mensagens do sistema.

- clique do mouse;
- pressionar tecla;
- chegada de mensagem;
- interação com interface gráfica.

Exemplos:

- JavaScript;
- C# (interfaces gráficas);
- Aplicações web;

```c
#include <stdio.h>

int main() {
    int x = 5;
    int resultado;

    resultado = x * x;

    printf("%d\n", resultado);
    return 0;
}
```

Código 3: Exemplo em linguagem C.

```haskell
quadrado x = x * x

main = print (quadrado 5)
```

Código 4: Exemplo em linguagem Haskell.

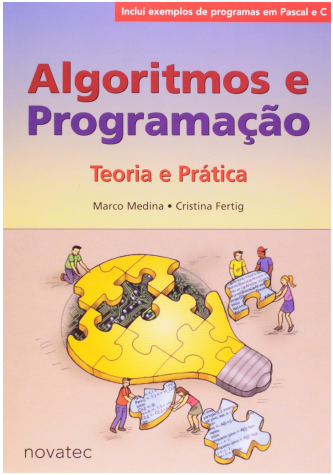
```java
class Numero {
    int valor;
    Numero(int v) {
        valor = v;
    }
    int quadrado() {
        return valor * valor;
    }
}
public class Main {
    public static void main(String[] args) {
        Numero n = new Numero(5);
        System.out.println(n.quadrado());
    }
}
```

Código 5: Exemplo em linguagem Java.

```
1 document.getElementById("botao").addEventListener("click",
  ↪ function() {
2     let x = 5;
3     let resultado = x * x;
4     document.getElementById("saida").innerText = resultado;
5 });
```

Código 6: Exemplo em linguagem Java Script.

(Medina; Fertig, 2005)



APPLE. **Xcode**. Acesso em: 8 jan. 2025. 2025. Disponível em: https://developer.apple.com/xcode.

DE OLIVEIRA, J.F.; MANZANO, J.A.N.G. **Algoritmos: Lógica para Desenvolvimento de Programação de Computadores**. 16. ed. São Paulo: Editora Érica, 2004.

DEITEL, Paul; DEITEL, Harvey. **C: Como Programar**. 6. ed. São Paulo: Pearson Universidades, 2011.

FOUNDATION, Eclipse. **Eclipse IDE**. Acesso em: 8 jan. 2025. 2025. Disponível em: https://www.eclipse.org.

FREE SOFTWARE FOUNDATION. **O Projeto GNU**. Acesso em: 16 mar. 2026. Projeto GNU. 2025. Disponível em: https://www.gnu.org/gnu/thegnuproject.pt-br.html.

GITHUB. **Atom - A hackable text editor for the 21st Century**. Acesso em: 5 jan. 2025. 2024. Disponível em: https://github.com/atom/atom.

GOOGLE. **Android Studio**. Acesso em: 8 jan. 2025. 2025. Disponível em: https://developer.android.com/studio.

JETBRAINS. **IntelliJ IDEA**. Acesso em: 8 jan. 2025. 2025. Disponível em: https://www.jetbrains.com/idea.

JETBRAINS. **PyCharm**. Acesso em: 8 jan. 2025. 2025. Disponível em: https://www.jetbrains.com/pycharm.

MEDINA, M.; FERTIG, C. **Algoritmos e Programação – Teoria e Prática**. São Paulo: Novatec, 2005.

MICROSOFT. **Visual Studio Code**. Acesso em: 5 jan. 2025. 2024. Disponível em: https://code.visualstudio.com/.

MICROSOFT. **Visual Studio Professional**. Acesso em: 8 jan. 2025. 2025. Disponível em: https://visualstudio.microsoft.com/vs/professional/.

RSTUDIO. **RStudio**. Acesso em: 8 jan. 2025. 2025. Disponível em: https://posit.co.

SUBLIME. **Sublime Text**. Acesso em: 5 jan. 2025. 2024. Disponível em: https://www.sublimetext.com/.

VIM. **Vim - the ubiquitous text editor**. Acesso em: 5 jan. 2025. 2024. Disponível em: https://www.vim.org/.

APPLE. **Xcode**. Acesso em: 8 jan. 2025. 2025. Disponível em: https://developer.apple.com/xcode.

DE OLIVEIRA, J.F.; MANZANO, J.A.N.G. **Algoritmos: Lógica para Desenvolvimento de Programação de Computadores**. 16. ed. São Paulo: Editora Érica, 2004.

DEITEL, Paul; DEITEL, Harvey. **C: Como Programar**. 6. ed. São Paulo: Pearson Universidades, 2011.

FOUNDATION, Eclipse. **Eclipse IDE**. Acesso em: 8 jan. 2025. 2025. Disponível em: https://www.eclipse.org.

FREE SOFTWARE FOUNDATION. **O Projeto GNU**. Acesso em: 16 mar. 2026. Projeto GNU. 2025. Disponível em: https://www.gnu.org/gnu/thegnuproject.pt-br.html.

GITHUB. **Atom - A hackable text editor for the 21st Century**. Acesso em: 5 jan. 2025. 2024. Disponível em: https://github.com/atom/atom.

GOOGLE. **Android Studio**. Acesso em: 8 jan. 2025. 2025. Disponível em: https://developer.android.com/studio.

JETBRAINS. **IntelliJ IDEA**. Acesso em: 8 jan. 2025. 2025. Disponível em: https://www.jetbrains.com/idea.

JETBRAINS. **PyCharm**. Acesso em: 8 jan. 2025. 2025. Disponível em: https://www.jetbrains.com/pycharm.

MEDINA, M.; FERTIG, C. **Algoritmos e Programação – Teoria e Prática**. São Paulo: Novatec, 2005.

MICROSOFT. **Visual Studio Code**. Acesso em: 5 jan. 2025. 2024. Disponível em: https://code.visualstudio.com/.

MICROSOFT. **Visual Studio Professional**. Acesso em: 8 jan. 2025. 2025. Disponível em: https://visualstudio.microsoft.com/vs/professional/.

RSTUDIO. **RStudio**. Acesso em: 8 jan. 2025. 2025. Disponível em: https://posit.co.

SUBLIME. **Sublime Text**. Acesso em: 5 jan. 2025. 2024. Disponível em: https://www.sublimetext.com/.

VIM. **Vim - the ubiquitous text editor**. Acesso em: 5 jan. 2025. 2024. Disponível em: https://www.vim.org/.

Estes slides estão protegidos por uma licença Creative Commons

Este modelo foi adaptado de Maxime Chupin.

Marisangila Alves, MSc
marisangila.alves@edu.sc.senai.br
marisangila.com.br

2026/1

Lógica de Programação

Liguagens de Programação