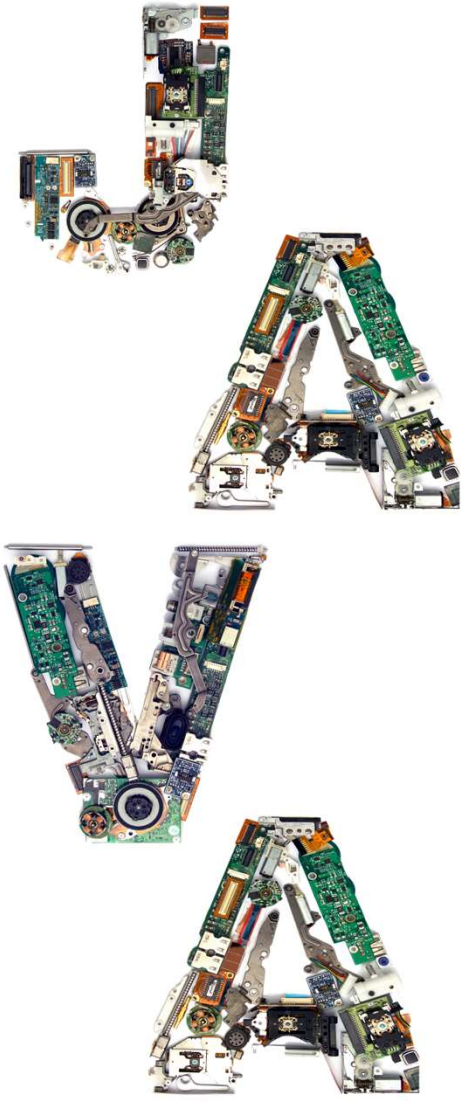
Programação Orientada a Objetos com Java e WEB

Paradigmas de Linguagens de Programação



Conteúdo

Paradigma de Linguagens de Programação

1. Introdução
2. Papel das linguagens no desenvolvimento de software
3. Classificação das linguagens de programação
4. Interface em camadas de computadores
5. Curiosidades
6. Implementação de LPs
7. Paradigmas de LPs
8. Origem das LPs
9. Índice TIOBE
10. Exercício

Introdução

Uma **linguagem de programação** (LP) é uma ferramenta utilizada pelo profissional de computação para escrever programas.

Programa de computador → conjunto de instruções (escritas em uma linguagem de programação) a serem seguidas pelo computador para realizar um determinado processo.

Um algoritmo é um programa de computador? Não necessariamente!!
Um programa de computador é um algoritmo? Sim

Introdução

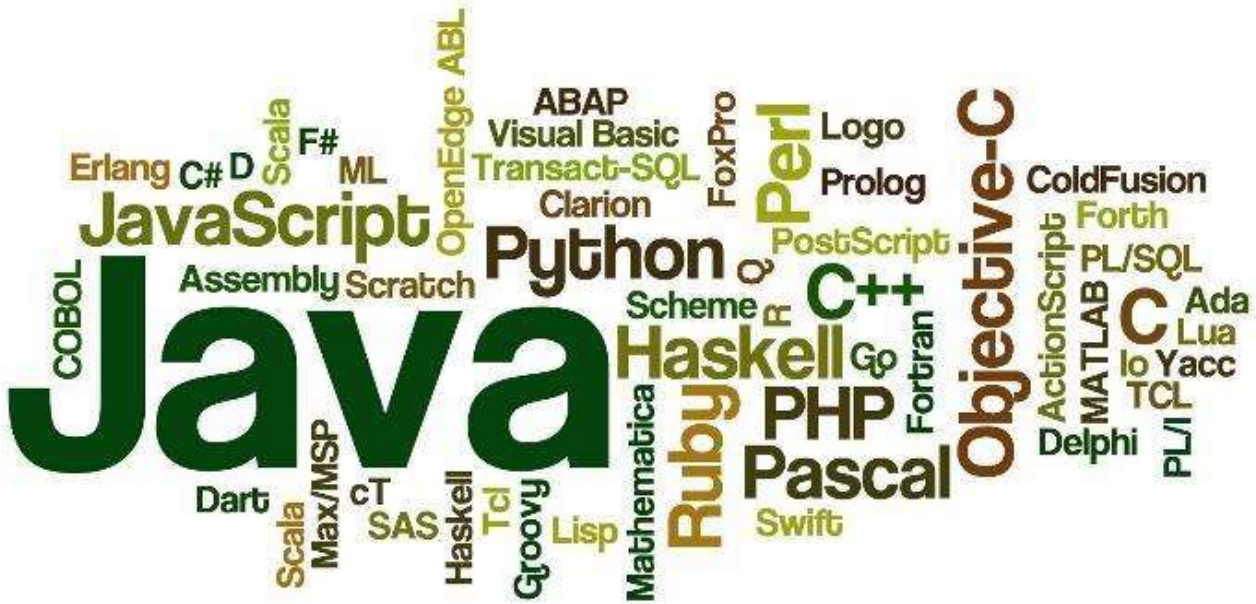
As LPs surgiram para substituir a programação em **linguagem de máquina**.

Linguagem de Máquina ⇒ linguagem específica, provocando baixa produtividade de software.

Razões para se conhecer LPs:

- Maior capacidade de desenvolver soluções computacionais para problemas.
- Maior habilidade ao usar uma LP.
- Maior capacidade para escolher LPs apropriadas.
- Maior habilidade para aprender novas LPs.
- *Maior habilidade para projetar novas LPs.*

Introdução



Qual a melhor linguagem de programação? Qual devo usar?
Qual devo aprender?

Papel das linguagens no desenvolvimento de software

Qual o papel das linguagens de programação no processo de desenvolvimento de software?

1

Tornar mais produtivo o trabalho dos programadores

2

Tornar mais efetivo o processo de desenvolvimento de software

3

Tornar mais produtiva a geração e a manutenção de software para garantir que ele seja produzido atendendo a padrões de qualidade

Papel das linguagens no desenvolvimento de software

Uma das principais **metas** das linguagens de programação é que programadores tenham maior **produtividade**, permitindo expressar suas intenções mais facilmente do que quando comparado com a linguagem que um computador entende nativamente (código de máquina)

Classificação das linguagens de programação

Segundo ACM (*Association for Computing Machinery*), as linguagens de programação podem ser classificadas de diversas formas. **Uma das classificações diz respeito ao grau de abstração**:

1. Linguagem de máquina
2. Linguagem de baixo nível
3. Linguagem de alto nível

Classificação das linguagens de programação

1

Linguagem de máquina

Todo computador possui um conjunto de instruções que seu processador é capaz de executar. Essas instruções, chamadas de código de máquina, são representadas por sequências de bits, normalmente limitadas pelo número de bits do registrador principal da CPU. **Esse código é chamado de código binário**. **São formados pelos bits 0 e 1**.

Classificação das linguagens de programação

2

Linguagem de baixo nível

São linguagens de programação que compreendem as características da arquitetura do computador. Assim, **utilizam somente instruções do processador**, para isso é necessário conhecer os registradores da máquina. Nesse sentido, as linguagens de baixo nível estão diretamente relacionadas com a arquitetura do computador. Um exemplo é a linguagem **assembly** (**não assembler**) que trabalha diretamente com os registradores do processador, manipulando dados.

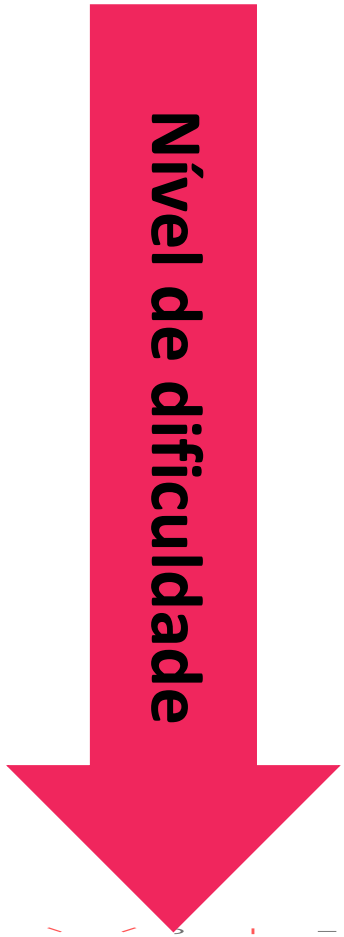
Classificação das linguagens de programação

3

Linguagem de alto nível

São linguagens com um nível de abstração relativamente elevado, **longe do código de máquina e mais próximo à linguagem humana**. As linguagens de alto nível não estão diretamente relacionadas à arquitetura do computador. O programador de uma linguagem de alto nível não precisa conhecer características do processador, como instruções e registradores. Essas características são abstraídas na linguagem de alto nível.

Classificação das linguagens de programação



Linguagem de Alto nível
- Próxima à linguagem humana.
- Portável.

Linguagem Assembly
- Representação de termos originários do inglês.
- Dependente da máquina.

Linguagem de Máquina
- Código binário (0s e 1s).
- Dependente da máquina.

Hardware do computador (“a máquina”)
- CPU.
- Memória (RAM, ROM, etc).
- Drives de disco.
- Entrada/saída.

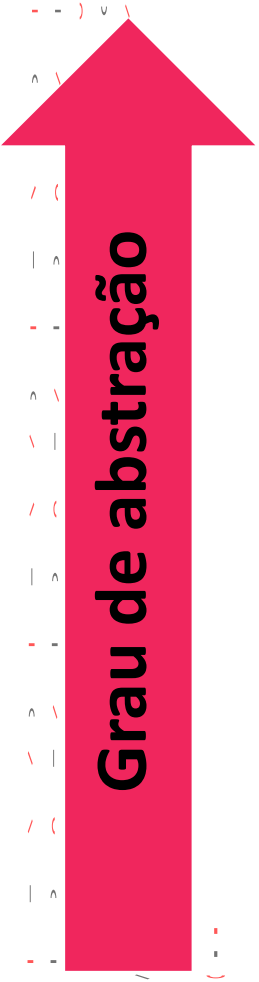
Sistemas de Informação | FIAP
Prof. Dr. Antonio Marcos SELMINI – selmini@fiap.com.br

Classificação das linguagens de programação

As linguagens de programação também podem ser classificadas em gerações, o que pode haver divergência de um autor para outro. As linguagens podem ser classificadas em 5 gerações:

1a	linguagem de máquina ou código de máquina
2a	linguagem de montagem (assembly)
3a	(linguagens procedurais) introdução dos tipos de dados, variáveis, estruturas de controles. Exemplo: C, Pascal, Algol, etc...
4a	linguagens orientada a objetos e linguagens de manipulação de dados (SQL). Exemplo: Java, C#, Phyton, etc...
5a	linguagens voltadas para inteligência artificial (PROLOG) e as linguagens funcionais (LISP)

Classificação das linguagens de programação



- 5ª geração: linguagens orientadas ao conhecimento (inteligência artificial)
- 4ª geração: linguagens orientadas à aplicação e orientadas a objetos
  - `select * from aluno;`
- 3ª geração: linguagens procedurais (orientadas ao usuário)
  - `int soma = x1 + y1;`
  - `if(soma >= 6) printf("Parabéns!!");`

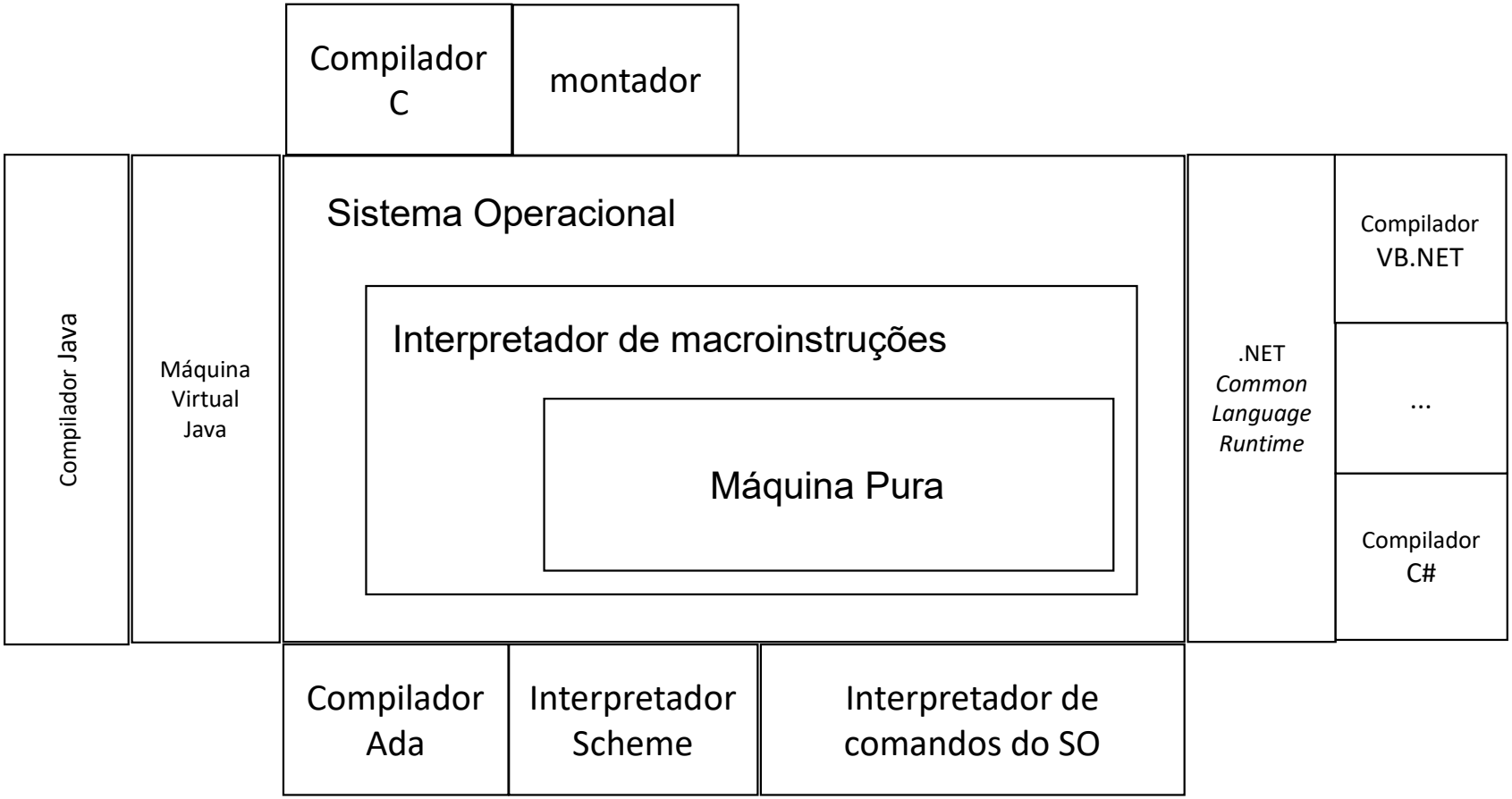
Alto nível

- 2ª geração: linguagem de montagem (*assembly*)
  - `ADD R1, TOTAL`
- 1ª geração: linguagem de máquina
  - `0010 0000 1100 1111`

Baixo nível

Prof. Dr. Antonio Marcos SELMINI – selmini@fiap.com.br

Interface em camadas das linguagens de programação



Curiosidades: você sabia?

O primeiro programa de computador foi criada por ***Ada Lovelace***, filha do escritor inglês ***Lord Byron***. Foi utilizado na calculadora programável criada por ***Charles Babbage.***

Uma das primeiras linguagens de programação para computadores foi ***Plankalkül***, criada na Alemanha.

A primeira linguagem de alto nível amplamente usada foi ***Fortan*** (criada em 1954).

Curiosidades: você sabia?

O primeiro compilador foi escrito por ***Grace Hopper*** em 1952 para a linguagem de programação ***A-0.***

Em 1957 foi criada ***B-0***, sucessora de ***A-0***, que daria origem a ***Flow-Matic*** (1958), antecessora imediata de ***Cobol***, de 1959.

A linguagem de programação ***Simula 67*** introduz o conceito de classes.

Curiosidades: você sabia?

Smalltalk expande o conceito de classes e se torna a primeira linguagem de programação que oferecia suporte completo à programação orientada a objetos.

A linguagem de programação **C++** (originalmente conhecida como C com classes) populariza a orientação a objetos.

Curiosidades: você sabia?

Katie Bouman é a criadora do algoritmo que captou e organizou as imagens do fenômeno.

Na época com 29 anos, **Katherine Bouman** trabalhava como pesquisadora de pós-doutorado no Centro de Harvard-Smithsonian de Astrofísica, ela é responsável pela liderança de uma equipe de mais de 200 cientistas.

Denominado "**Event Horizon Telescope Collaboration**", o time tinha o objetivo de coletar dados e analisar a captura de imagens de diversos telescópios espalhados pelo mundo, todos envolvidos na análise da estrutura do buraco negro.

Fonte: https://claudia.abril.com.br/noticias/quem-e-a-mulher-por-tras-da-1a-foto-de-buraco-negro-da-historia/. Acesso em 01/02/2022.

Curiosidades: você sabia?

A cientista da computação do MIT **Katie Bouman** com pilhas de discos rígidos de dados de imagens de buracos negros.

A cientista da computação do MIT **Margaret Hamilton** com o código que ela escreveu que ajudou a colocar um homem na lua.

Sugestão de filme para assistir

No auge da corrida espacial travada entre Estados Unidos e Rússia durante a Guerra Fria, uma equipe de cientistas da NASA, formada exclusivamente por mulheres afro-americanas, provou ser o elemento crucial que faltava na equação para a vitória dos Estados Unidos, liderando uma das maiores operações tecnológicas registradas na história americana e se tornando verdadeiras heroínas da nação.

Trailer oficial: https://www.youtube.com/watch?v=2CIqexd838s

Fonte: https://g.co/kgs/1BX5Ug. Acesso em 01/02/2022.

Implementação de Linguagens de Programação

Todo programa escrito em uma LP deve ser traduzido para a linguagem de máquina para ser executado.

Isso é realizado através de um programa ou conjunto de programas.

Esse programa “tradutor” recebe como entrada o **código fonte** e gera o código de máquina correspondente.

Existem três maneiras de traduzir o código fonte:

1	2	3
Compilação	Interpretação	Híbrido

Implementação de Linguagens de Programação

1 Compilação

Efetua a tradução integral do código fonte para o código de máquina.

A **execução é mais rápida** porque não é necessário fazer nenhuma tradução intermediária.

Para que o programa seja executado é necessário apenas o código executável.

A partir do código executável, o programa pode ser executado diversas vezes sem a necessidade de recompilação.

Sistemas de Informação | FIAP
Prof. Dr. Antonio Marcos SELMINI – selmini@fiap.com.br

1 Compilação

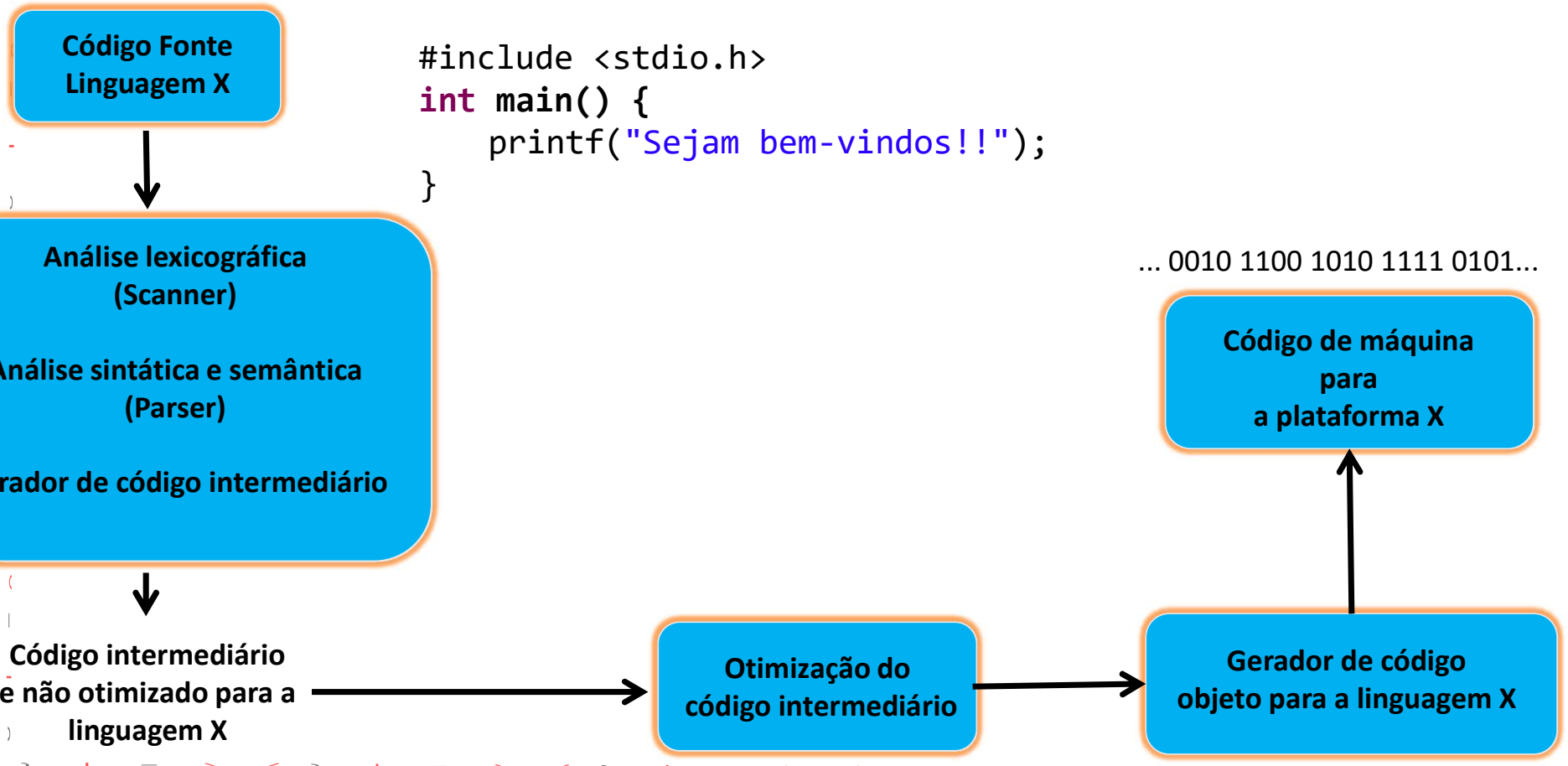
A **desvantagem** é a **não portabilidade** do código executável.

Não há depuração, pois o código executável não guarda referência do código fonte.

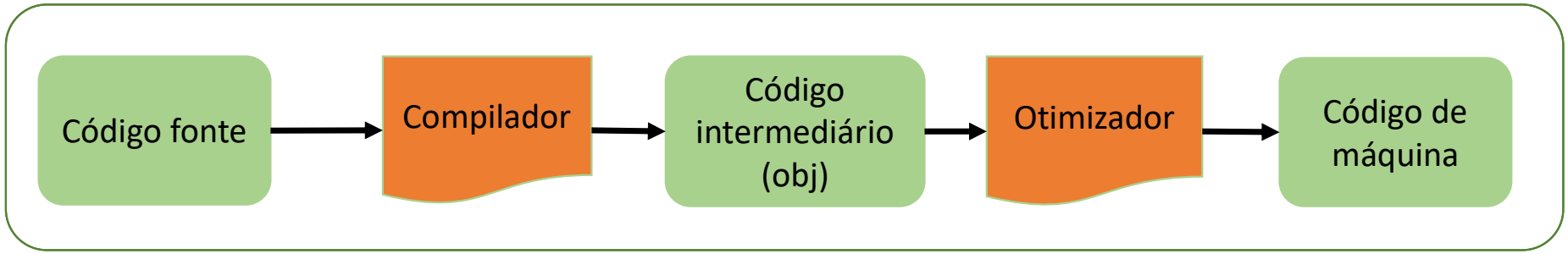
O código gerado é executado pelo sistema operacional.

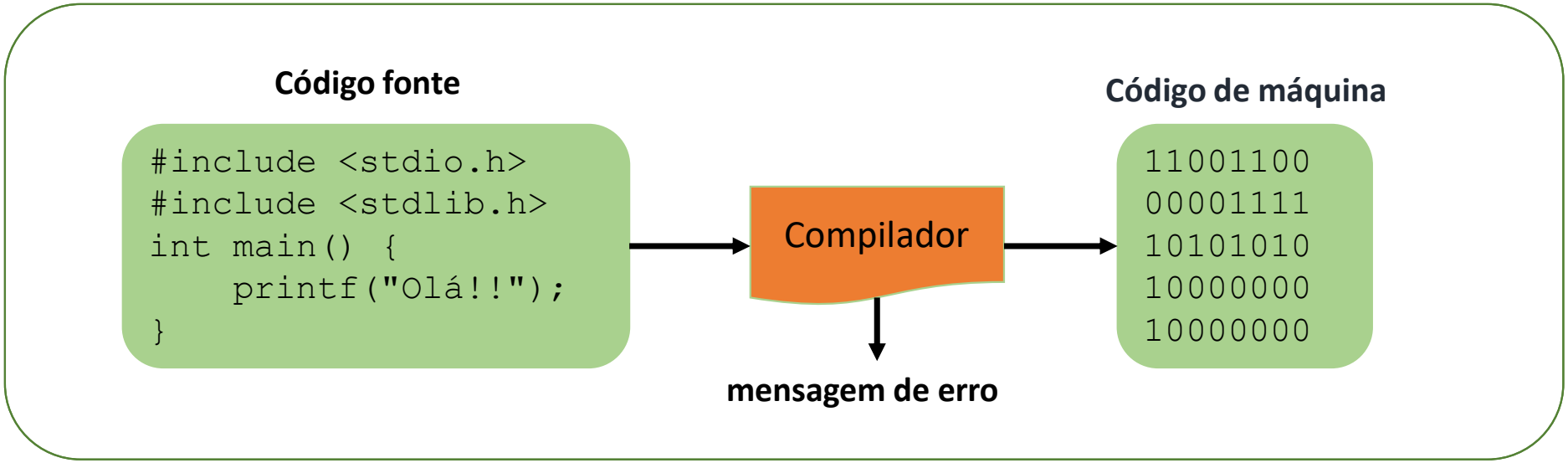
Exemplos: C, C++ e Pascal.

Implementação de Linguagens de Programação



Implementação de Linguagens de Programação





Implementação de Linguagens de Programação

2 Interpretação

Programa interpretador “entende” as instruções escritas em uma LP.

No momento da execução, a instrução é traduzida para linguagem de máquina e executada.

Vantagem: partes do programa podem ser executados e há a **portabilidade** de código.

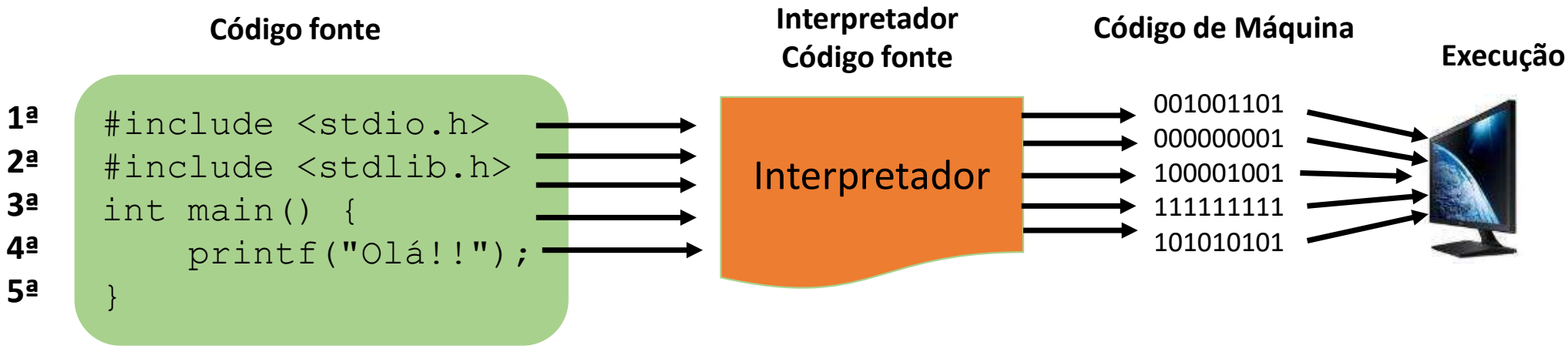
O código gerado pode ser executado pelo sistema operacional (python) ou pode ser interpretado por navegadores para as linguagens WEB (javascript)**.**

Implementação de Linguagens de Programação

2 Interpretação

- **Desvantagens**:
  - processo lento em relação ao processo de compilação.
  - Maior consumo de memória, pois há a necessidade de armazenar o código-fonte, tabela de símbolos e o programa interpretador.
  - A lentidão está no processo de decodificação das sentenças em linguagem de máquina.
  - Em relação às linguagens compiladas, a tradução e a execução de programas interpretados podem ser vistos como um único processo.
- Exemplos: JavaScript, BASIC, Python, Perl, Ruby, etc...

Implementação de Linguagens de Programação



Cada linha do código fonte (uma linha de cada vez) é interpretada e executada

Implementação de Linguagens de Programação

Compilação x Interpretação → Ambos são tradutores de código

Compilador gera executável	**Interpretador não gera executável**
Compilador analisa a gramática e a sintaxe da linguagem e aponta os erros. O executável é gerado depois que todos os erros forem corrigidos	**Interpretador interpreta as linhas de código e não mostra erros**

Implementação de Linguagens de Programação

3 Híbrido

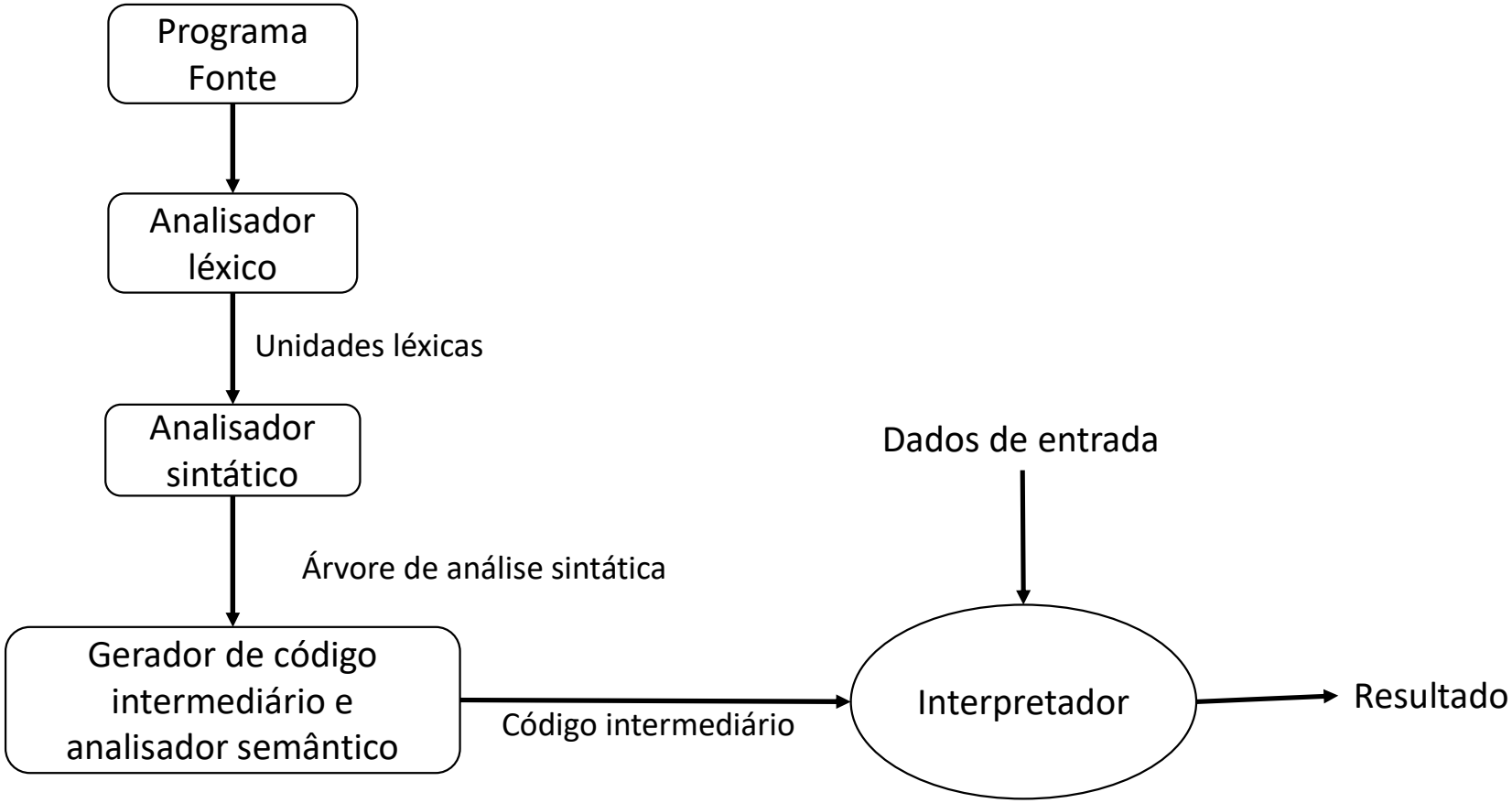
Combina a **execução eficiente** (**compilação**) e a **portabilidade de programas** (**interpretação**).

A base é a existência de um código intermediário, mais fácil de ser interpretado e não específico de uma plataforma computacional (sistema operacional).

Método dividido em duas etapas: compilação para um código intermediário e interpretação desse código.

Nesta categoria podemos citar o Java e o C#. O código Java é compilado para a plataforma Java (*bytecodes*), enquanto o código C# é compilado para a plataforma CLI (*Common Language Infrastructure*).

Implementação de Linguagens de Programação



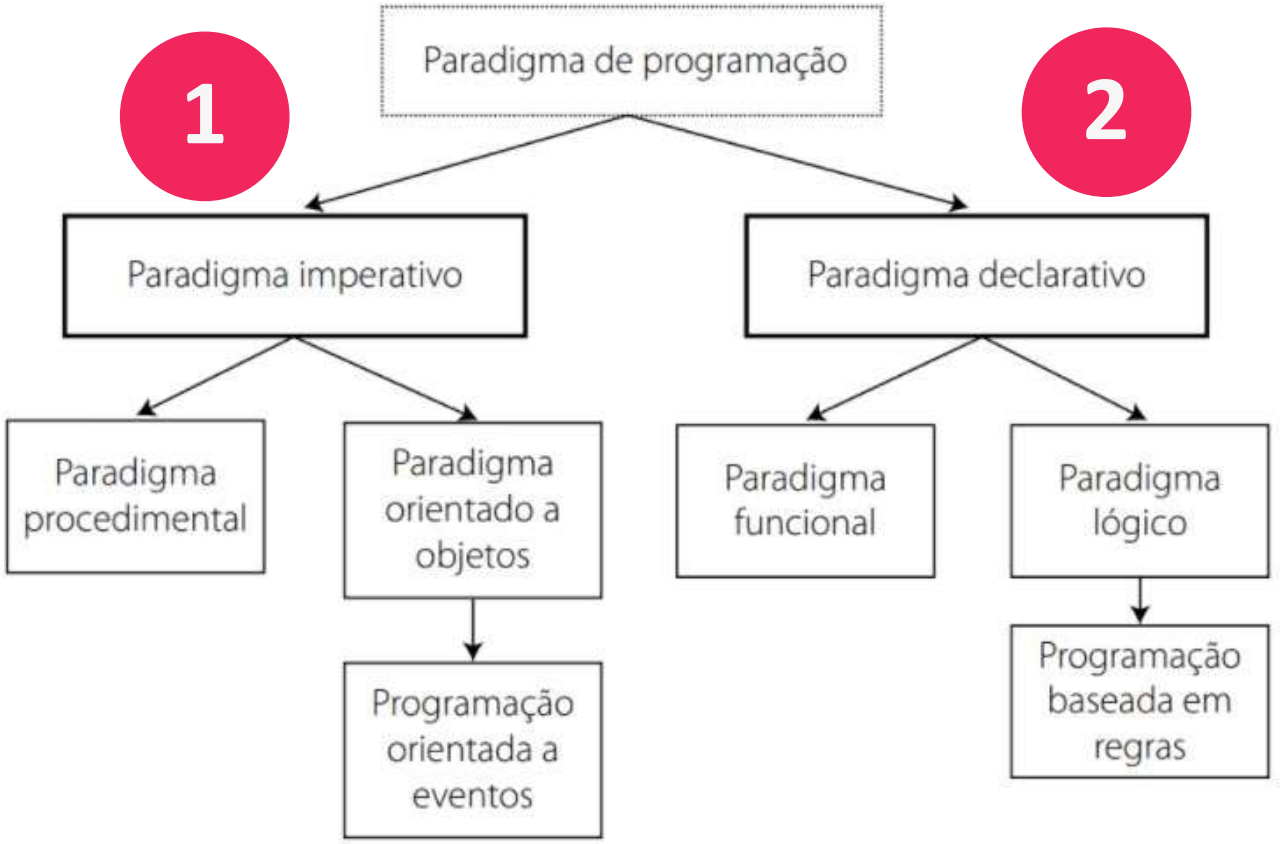
Paradigmas de Linguagens de Programação

Toda linguagem de programação está construída sobre um paradigma.

Um **paradigma** representa um padrão de pensamento que guia um conjunto de atividades relacionadas. Trata-se de um padrão que define um modelo para a resolução de problemas e regra, basicamente, toda linguagem de programação. Os paradigmas de programação estão classificados em quatro diferentes tipos, que evoluíram ao longo dos anos: **paradigma imperativo**, **paradigma funcional**, **paradigma lógico** e **paradigma orientado a objetos**.

Os tipos de programação estão diretamente ligados ao tipo de paradigma na qual a linguagem de programação foi concebida.

Paradigmas de Linguagens de Programação



Fonte: Adaptado de SIMÃO, J. M. Orientação a objetos: programação em C++. Curitiba: Departamento Acadêmico de Eletrotécnica, Universidade Federal Tecnológica do Paraná, 2018. 26 p. (Notas de aula). Disponível em: http://www.dainf.ct.utfpr.edu.br/~jeansimao/Fundamentos1/ LinguagemC++/Fundamentos1-2-SlidesC++1-A-2018-08-01.pdf. Acesso em: 25 ago. 2019.

Paradigmas de Linguagens de Programação

1

Imperativo

Fundamentado na ideia de computação como um processo que realiza mudanças de **estados** (**variáveis**). Representa uma sequência de comandos para o computador executar. O programador "diz" ao computador: faça isso, depois isso, depois aquilo...

Um estado representa uma configuração (valor ou conjunto de valores) da memória do computador.

LPs incluídas nesse paradigma **especificam como uma computação é realizada por uma sequência** de alterações no estado da memória do computador.

Podemos relacionar com o comportamento imperativo das linguagens naturais que expressam ordens.

Paradigmas de Linguagens de Programação

1

Imperativo

O foco dos programas é especificar como um processamento deve ser feito no computador.

Elementos da programação imperativa: definição de tipos de dados, expressões e atribuições, estrutura de controle de fluxo e definições de sub-rotinas.

Esse paradigma é subdividido em: **estruturado** (ou procedural), **orientado a objetos** e **concorrente.**

Paradigmas de Linguagens de Programação

1.1

Imperativo Estruturado

Baseado na ideia de desenvolvimento de programas por refinamento sucessivos (*top-down*).

Sugere que todos os programas possíveis podem ser reduzidos a apenas três estruturas: **sequência**, **decisão ou seleção** e **iteração ou repetição.**

Também pode ser chamada de **programação modular** ou **procedimental**, onde são utilizados nos programas sub-rotinas (**procedimentos** / **funções** / **métodos**).

Não utilizam desvios incondicionais (*goto*).

Exemplos de LPs: Pascal, C, etc.

Paradigmas de Linguagens de Programação

1.1

Imperativo Estruturado

```c
#include <stdio.h>
#include <math.h>
int lerNum();
int raiz(int z);
int main(void) {
    int x, r;
    x = lerNum();
    r = raiz(x);
}
int lerNum() {
    int y;
    scanf("%d", &y);
    return y;
}
int raiz(int z) {
    return sqrt(z);
}
```

Linguagem C

```pascal
program pares;
var x, y: integer;
begin
 writeln('Digite os dois valores');
 readln(x, y);
 if (x mod 2)<> 0 then
    x := x + 1;
    while x<= y do
    begin
       (x, ' - '); x := x + 2;
    end;
    ('Fim da Lista');
end.
```

Linguagem Pascal

Sistemas de Informação | FIAP
Prof. Dr. Antonio Marcos SELMINI – selmini@fiap.com.br

Paradigmas de Linguagens de Programação

1.2

Imperativo Orientado a Objetos

Neste tipo de paradigma, o programa é entendido como um conjunto de objetos que interagem entre si.

Objetiva tornar mais rápido e confiável o desenvolvimento de sistemas, focando os dados como elementos básicos.

Os conceitos importantes são: **classes**, **objetos**, **herança** e **polimorfismo.**

O paradigma orientado a objetos pode ser considerado uma evolução do paradigma estruturado.

Exemplo de LPs: C++, Java, C#, SmallTalk, Simula.

Paradigmas de Linguagens de Programação

1.2

Imperativo Orientado a Objetos

```java
public class Pessoa {
    String nome;
    int idade;

    public Pessoa(String nome, int idade) {
        this.nome = nome;
        this.idade = idade;
    }
}
```

```java
public class Aluno extends Pessoa {
    int rm;

    public Aluno(String nome, int idade, int rm) {
        super(nome, idade);
        this.rm = rm;
    }
}
```

Sistemas de Informação | FIAP
Prof. Dr. Antonio Marcos SELMINI – selmini@fiap.com.br

Paradigmas de Linguagens de Programação

2

Declarativo

Neste paradigma, os programas são especificações de como é a tarefa a ser realizada e não como seus procedimentos funcionam.

Linguagens de marcação são declarativas, pois descrevem o que são suas estruturas e não como elas serão utilizadas.

O programador não precisa saber como o computador é implementado (arquitetura) , nem sobre a maneira pelo qual ele é melhor utilizado.

Os programas são especificações de relações e funções.

Não existem atribuições as variáveis. As variáveis são incógnitas e não representam posições de memória.

Paradigmas de Linguagens de Programação

2.1

Declarativo Funcional

LPs funcionais operam apenas com funções (**métodos**), tratando a computação como uma avaliação de funções matemáticas, evitando estados (variáveis).

As funções recebem listas de valores e retornam um valor como resposta do problema (objetivo da programação).

Um programa funcional é uma chamada de função (métodos) que chama outras funções (**processo recursivo**).

As principais operações são a definição de funções e suas chamadas recursivas.

São utilizadas mais no meio acadêmico do que no desenvolvimento commercial de software. Exemplo de LPs: Lisp, Haskell e ML.

Paradigmas de Linguagens de Programação

2.1

Declarativo Funcional

Paradigma funcional

```
(defun fatorial (n)
   (if (<= n1)
      1
   (* n (fatorial (- n1)))))
```

Paradigma Imperativo Estruturado (procedimental)

```
int fatorial(int n) {
   int fat = 1, cont;
   for(cont = 1; cont <= n; cont++) {
      fat = fat * cont;
   }
   return fat;
}
```

Sistemas de Informação | FIAP
Prof. Dr. Antonio Marcos SELMINI – selmini@fiap.com.br

Paradigmas de Linguagens de Programação

2.2 Declarativo Lógico

É um paradigma de programação que faz uso da **lógica matemática**. **São baseadas em cálculo de predicados**.

Um predicado define uma relação entre constantes ou variáveis.

A execução dos programas corresponde a um processo de dedução automático.

Exemplo de LP: Prolog

```
pai(jose, ana);
pai(pedro, jose);
avo(x, z) :- pai(x, y), pai(y, z);
?-avo(x, ana).
```

Paradigmas de Linguagens de Programação

Como interpretar as descrições?

Python é uma linguagem de programação de alto nível, interpretada, de script, imperativa, orientada a objetos, funcional, de tipagem dinâmica e forte. Foi lançada por Guido van Rossum em 1991.

Disponível em: https://pt.wikipedia.org/wiki/Python. Acesso em 31/01/2022.

C# é uma linguagem de programação, multiparadigma, de tipagem forte, desenvolvida pela Microsoft como parte da plataforma .NET. A sua sintaxe orientada a objetos foi baseada no C++ mas inclui muitas influências de outras linguagens de programação, como Object Pascal e, principalmente, Java. O código fonte é compilado para **Common Intermediate Language** (CIL) que é interpretado pela máquina virtual Commom Language Runtime (CLR).

Disponível em: https://pt.wikipedia.org/wiki/C_Sharp. Acesso em 31/01/2022.

Origem das Linguagens de Programação

- Fortan (1957).
- Lisp (1959).
- Algol (1960).
- Cobol (1960).
- Basic (1964).
- Pascal (1971).
- C (1972).
- Prolog (1972).
- SmallTalk (1972).
- Ada (1983).
- C++ (1985).
- Python (1991).
- Java (1995).

Índice TIOBE

http://tiobe.com/tiobe-index/

Exercício (ENADE 2014)

Na figura 1, abaixo, está representado, esquematicamente, um processo de tradução de um programa (arquivo fonte) em código binário. Esse processo de compilação clássica é utilizado em compiladores como os das linguagens C e Pascal.

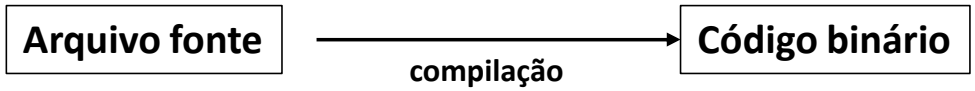


Figura 1

Na figura 2, abaixo, está representado, esquematicamente, um processo de tradução de um programa (arquivo fonte) em código intermediário. Esse processo híbrido é utilizado em compiladores como os das linguagens Java e C#.

Figura 2

Exercício (ENADE 2014)

Considerando que, em ambos os processos, o código binário é o que será executado pelo computador, e que a execução dos dois programas gerados, cada qual por um dos processos apresentados, ocorre em situações equivalentes, avalie as afirmações a seguir.

I) Há portabilidade para a execução de ambos os programas gerados em cada processo.

II) Na execução do programa gerado por meio do processo híbrido, o consumo de memória é maior que na execução pelo processo de compilação clássica.

III) O desempenho na execução do programa gerado pelo processo de compilação clássica é melhor que na execução pelo processo híbrido.

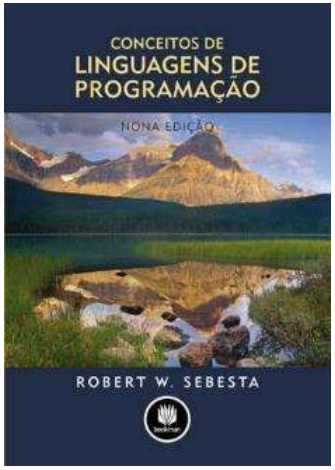
É correto o que se afirma em:

a) I, apenas.
b) III, apenas.
c) I e II, apenas.
d) II e III, apenas.
e) I, II e III.

Bibliografia

- SEBESTA, R. W. Conceitos de Linguagem de Programação, 9ª edição, Porto Alegre: Bookman, 2011.

- VAREJÃO, F. Linguagens de Programação: Conceitos e Técnicas. 1ª edição, Editora Campus, Rio de Janeiro, 2004.



Bibliografia

- MELO, Ana Cristina Vieira de; SILVA, Flávio Soares Corrêa da. *Princípios de Linguagens de Programação*. São Paulo: Edgard Blücher Ltda, 2003. 211 pp. p. 7-11.

- Sethi, Ravi. *Programming Languages: Concepts & Constructs*. 2ª. ed. Reading, Massachusetts: Addison-Wesley, 1996. 640 pp. p. 4-8.