Paradigmas de Linguagens de Programação

Aula 01

Prof. Marcelo de Gomensoro Malheiros

Paradigmas de Linguagens de Programação

Apresentação da disciplina

Objetivo

Estudar os principais conceitos sobre programação de computadores e os diversos paradigmas existentes

Pré-requisitos

Base de Programação: Algoritmos, Estruturas de Dados e Programação Orientada a Objetos

Conteúdo programático

1. Conceitos sobre programação de computadores. Propriedades desejáveis e especificação de uma Linguagem de Programação. Métodos de implementação. Paradigmas e evolução das Linguagens de programação.

2. Revisão de Java (amarrações, valores e tipos de dados, variáveis e constantes, expressões e comandos, modularização).

3. Orientação a objetos em Java (herança, polimorfismo, acesso protegido, interfaces).

4. Exceções, pacotes e genéricos em Java.

Conteúdo programático

5. Introdução a C/C++ (amarrações, valores e tipos de dados, variáveis e constantes, expressões e comandos, modularização).

6. Orientação a objetos em C++ (herança, polimorfismo, acesso protegido). Sobrecarga de operadores e exceções em C++.

7. Outras linguagens imperativas (linguagens de *script*, Pascal, Delphi, Visual Basic, C#, D, Objective C, Eiffel).

Conteúdo programático

8. Paradigma funcional (Lisp, Scheme, Haskell).

9. Paradigma lógico (Prolog).

10. Outros paradigmas. Amarração entre linguagens.

11. Linguagens esotéricas.

12. Linguagens de *script* (Python, PHP, Perl, Ruby).

Forma de avaliação

- Projetos de implementação, em diferentes linguagens
- Atividades práticas em laboratório
- Participação na aula e cumprimento dos prazos

Bibliografia básica

LUTZ, M.; TORTELLO, J. **Aprendendo python.** Porto Alegre: Bookman, 2008.

SEBESTA, R. **Conceitos de linguagens de programação.** Porto Alegre: Bookman, 2003.

VAREJÃO, F. **Linguagens de Programação:** Java, C e C++ e outras - conceitos e técnicas. Rio de Janeiro: Campus, 2004.

Bibliografia complementar

DEITEL, H. M.; DEITEL, P. J. **Java:** como programar. Bookman, 2003.

FLANAGAN, David. **Java:** o guia essencial. Rio de Janeiro: Campus, 2000.

JANDL JUNIOR, Peter. **Introdução ao Java.** São Paulo: Berkeley, 2002.

MIZRAHI, V. V. **Treinamento em linguagem C++.** Makron, 1995.

MORAIS, P.; PIRES, J. N. **Python:** curso completo. Lisboa: FCA, 2002.

PILGRIM, M. **Mergulhando no Python.** Alta Books, 2005.

SANTOS, R. **Introdução à programação orientada a objetos usando Java.** Rio de Janeiro: Campus, 2003.

SCHILDT, H. **C completo e total.** São Paulo: Makron Books, 1997.

Código de cooperação

- Assista a todos os encontros e seja pontual
- Preste atenção; evite comportamento dispersivo
- Não deixe as dúvidas se acumularem; faça perguntas sempre que necessário
- Cumpra os seus compromissos a tempo
- Evite conversações laterais

Paradigmas de Linguagens de Programação

Conceitos básicos

Linguagens de Computador

Linguagens de Programação

×

Linguagens de Marcação

×

Linguagens de Consulta

Linguagens de Programação

- O que são LPs?
- Vale a pena estudá-las?
- Preciso conhecer todas as LPs?
- O foco será em LPs de alto ou baixo nível?
- Quais tipos de LPs existem?
- Quais LPs preciso aprender?
- Qual LP devo utilizar em um problema?
- Qual é a melhor LP?

Linguagens de Programação

Razões para estudar LPs:

- Maior capacidade de solução de problemas
- Maior habilidade no uso de uma dada LP
- Maior capacidade de selecionar a LP adequada
- Maior habilidade para aprender novas LPs
- Maior capacidade para projetar novas LPs

Linguagens de Programação

Três propriedades requeridas de um software:

- Confiabilidade

  Integridade e segurança contra erros de programação e do usuário

- Manutenibilidade

  Facilidade de alteração de um software

- Eficiência

  Uso otimizado dos recursos computacionais

Linguagens de Programação

Nove propriedades desejadas de uma LP:

- Legibilidade

  Facilidade de ler e entender um programa
- Redigibilidade

  Facilidade de escrever um programa
- Confiabilidade

  Incentivo à criação de programas confiáveis
- Eficiência

  Facilidade de criar programas eficientes

Linguagens de Programação

- Facilidade de aprendizado

  O quão fácil é dominar os recursos de uma LP
- Ortogonalidade

  Consistência e coerência de uso da LP
- Reusabilidade

  Capacidade de gerar código reutilizável
- Modificabilidade

  Facilidade em efetuar modificações no programa
- Portabilidade

  Comportamento independente de arquitetura

Especificação de LPs

Notação BNF (Backus-Naur Form):

Gramática que fornece a descrição léxica e sintática de uma LP

Exemplo:

<programa> → begin <lista_inst> end

<lista_inst> → <inst> | <inst> ; <lista_inst>

<inst> → <var> = <expressão>

<var> → A | B | C

<expressão> → <var> | <var> + <var> |
<var> - <var>

Especificação de LPs

Notação BNF (Backus-Naur Form) estendida:

Utiliza as construções (), [] e {}, a primeira denotando escolha, a segunda opcionalidade e a terceira, repetição

Exemplo:

<seleção> → if (<expressão>) <instrução>
[else <instrução>] ;

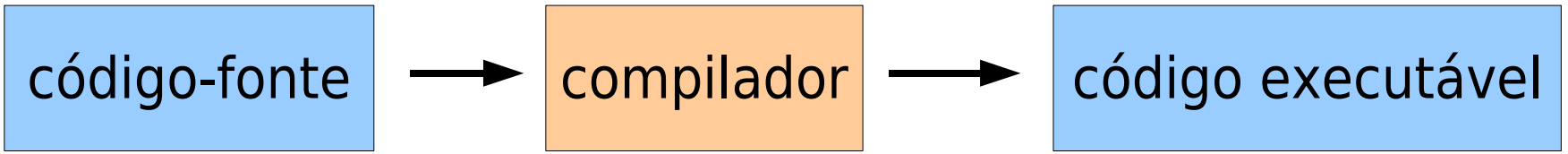
<lista_var> → <var> { ; <var> }

<var> → A | B | C

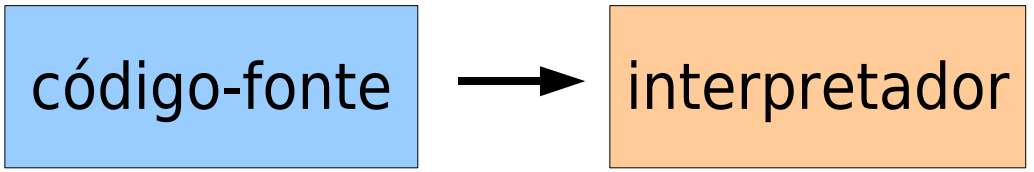
<expressão> → <var> | <var> (+ | -) <var>

Métodos de Implementação de LPs

- Compilação
  - geração de código executável eficiente

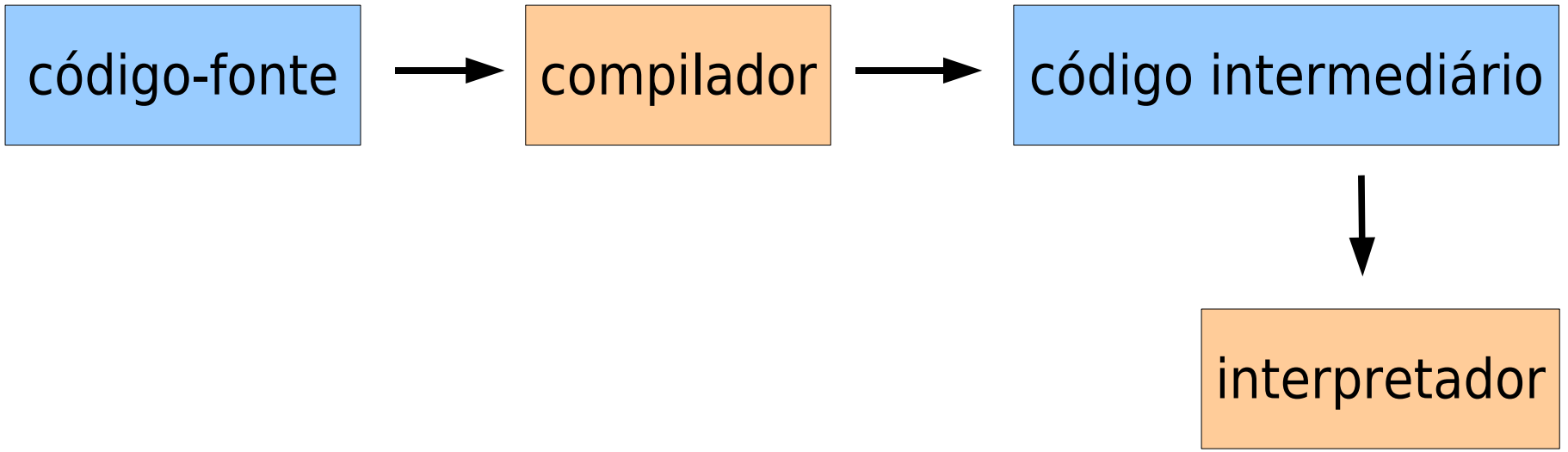


- Interpretação
  - sem geração de código executável

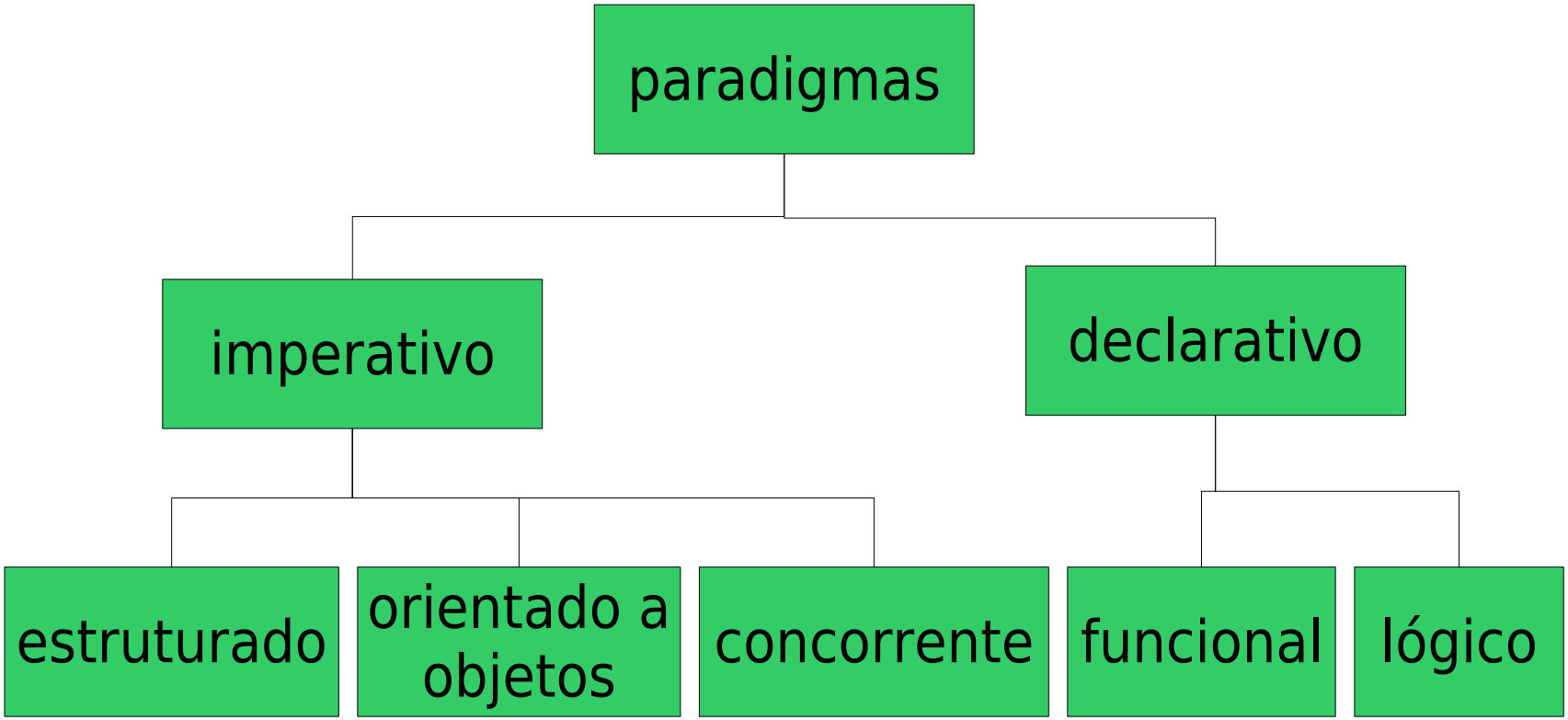


Métodos de Implementação de LPs

- Híbrido (compilação + interpretação)
  - geração de código executável portável, que por sua vez é interpretado por uma Máquina Virtual



Paradigmas de LPs



Paradigmas de Linguagens de Programação

Evolução das LPs

Início

- Ada Lovelace (1815-1852): primeira programadora (conceitual, da máquina de Babbage)
- Alan Turing (1912-1954): pioneiro na teoria das máquinas universais
- John von Neumann (1903-1957): propôs o modelo de arquitetura para máquinas eletrônicas programáveis
- Anos 40: programação física através de fios e controles mecânicos nos primeiros computadores

Décadas de 50 e 60

- Primeiras linguagens de baixo nível
- Primeira linguagem de médio nível: FORTRAN (*FORmula TRANslator*): de 1957, para aplicações científicas

```
program hello
    print*, 'Hello, world!'
end
```

- ALGOL (*ALGOrithmic Language*): de 1958, científica

```
BEGIN
    printf(($"Hello, world!"l$))
END
```

Décadas de 50 e 60

- LISP (*LISt Processor*): de 1958, para aplicações de Inteligência Artificial, deu origem ao LOGO (uma versão simplificada)

```
(write-line "Hello, world!")
```

- Apoio do governo americano para criar uma linguagem comercial: COBOL (*COmmon Business Oriented Language*), de 1960, de grande aceitação e ainda utilizada atualmente

```
IDENTIFICATION DIVISION.
PROGRAM ID.  HELLO-WORLD.
PROCEDURE DIVISION.
    DISPLAY "Hello, world!"
    STOP RUN.
```

Décadas de 50 e 60

- BASIC (*Beginners All-purpose Symbolic Instruction Code*): de 1964, interpretada, orientada ao ensino de programação, tornou-se popular por ser fácil de aprender

```
10 PRINT "Hello, world!"
20 END
```

- SIMULA 67 (*Simulation Language*): de 1967, uma extensão da ALGOL, primeira linguagem a usar os conceitos de orientação a objetos

```
BEGIN
    OutText("Hello, world!");
    OutImage;
END
```

Décadas de 50 e 60

- Problemas com crescimento dos sistemas
  - lógica de programação muito livre (lógica “espaguete”)
  - necessidade de maior disciplina ao programar
  - falta possibilidade de reutilizar código já escrito

Década de 70

- Tendências em linguagens: estruturação e modularidade
- Pascal: de 1971, programação estruturada, didática e muito popular na academia

```
program hello;
begin
    writeln('Hello, world!');
end.
```

- Tendências nos sistemas: descentralização e distribuição
- Aparecimento da indústria de software: hardware e software vendidos separadamente pela primeira vez

Década de 70

- C: de 1974, linguagens estruturada e modular
  - produz código em linguagem de máquina bem compacto
  - uso em software básico
  - surgiu ligada ao UNIX

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

Década de 70

- Primeiro padrão de redes: SNA da IBM (1974)
- Primeiro sistema operacional (SO) para microcomputadores: CP/M (1975)
- Interpretador BASIC para o 8080: fundação da Microsoft
- Primeira planilha: VisiCalc (*VISIble CALCulator*), de 1978
  - Primeiro aplicativo de uso específico, próximo ao usuário
- Primeiro processador de texto: WordStar, de 1978

Década de 80

- BASIC vira a linguagem padrão dos microcomputadores
- Sistema operacional do IBM-PC: MS-DOS em 1980
- Ada: de 1980, avançada para seu tempo, padronizada pelo DOD americano, então o maior usuário de softwares
  - tentativa de linguagem universal, muito complexa

```
with TEXT_IO;
procedure HELLO is
begin
    TEXT_IO.PUT_LINE ("Hello, world!");
 end HELLO;
```

Década de 80

- Smalltalk: de 1980, populariza a orientação a objetos

  Transcript show: 'Hello, world!'
- Linguagens de quarta geração → acesso a bancos de dados:
  - Adabas/Natural (IBM)
  - SQL (*Structured Query Language*): deu início ao domínio dos Bancos de Dados relacionais
- Banco de dados: DBase II, de 1982
- Mac OS (1984): sistema operacional do Macintosh: interface gráfica com o usuário (GUI)
- Windows 1.0 (1985)

Década de 80

- Linguagem Perl (1987)

```
qq chop lc and print chr ord uc q chop uc and print chr ord q ne sin and
print chr ord qw q le q and print chr ord q else and print chr ord q pop
and print chr oct oct ord uc qw q bind q and print chr ord q q eq and print
chr ord qw q warn q and print chr ord q pop and print chr ord q qr q and
print chr ord q else and print chr ord qw q do q and print chr hex length
q q semctl setpgrp chop q
```

ou

```
''=~('(?{'.('[[).[|`%,,/`[/[@$'^'+)@@/^(@@@@@,@),@').'! "})')
```

ou ainda

```
print "Hello, world!\n"
```

Década de 80

- Minix (1987): Unix para computadores pessoais
- OS/2 (1987): aprimoramento do Windows para o 386
- C++ (1988): aprimoramento do C com conceitos de orientação a objetos

```
#include <iostream>

int main()
{
   std::cout << "Hello, world!\n";
}
```

Década de 90 e além

- Linguagem Python (1990)

```
print "Hello, world!"
```

- Linux (1991)
- Windows 3.1 (1992): GUI padrão para o PC
- OS/2 Warp e Windows NT (1993)
- Windows 95 (1995)
- Linguagens PHP e Delphi (1995)

```
<?php
    echo "Hello, world!";
?>
```

Década de 90 e além

- Java (1995): usa orientação a objetos, é a primeira linguagem de grande popularidade a combinar compilação e interpretação

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

- Mac OS X (1999)

Década de 90 e além

- C# (2001): diretamente inspirada em Java e C++, controlada pela Microsoft

```
class Hello
{
   static void Main()
   {
      System.Console.WriteLine("Hello, world!");
   }
}
```

- Recentemente: D, Scala, Go, Groovy, Dart, ...