Linguagens de Programação
Conceitos e Técnicas

Introdução

Prof. Aléssio M. Junior

2025/1

Motivação

- Já em 1969, SAMMET listou 120 LPs que eram relativamente bem difundidas. Desde então muitas centenas de outras LPs surgiram (e desapareceram).
- A maioria dos programadores, todavia, nunca chega a usar mais do que umas poucas linguagens. Muitos não usam mais que uma ou duas em sua inteira vida profissional.
- Por que então estudar linguagens de programação?

Razões para Estudar LPs

- Maior capacidade de desenvolver soluções computacionais para problemas (ex., uso eficiente de recursividade)
- Maior habilidade ao usar uma LP (ex., uso eficiente dos recursos de gerenciamento de memória)
- Maior habilidade escolher uma linguagem de programação por conhecer seus pontos fortes e fracos (++ depende muito do conhecimento dos programadores).

Razões para Estudar LPs

- Torna mais fácil aprender novas LPs e acompanhar a tecnologia (ex., quanto mais LPs você souber, mais fácil se torna aprender uma nova LP).
- Maior habilidade para aprender novas LPs
- Maior habilidade para projetar novas LPs
- Aumentar o seu vocabulário de construção úteis de
- programação

Leituras Interessantes

- *A vingança dos nerds*
  http://www.paulgraham.com/icad.html
- *A linguagem de cem anos*
  http://www.paulgraham.com/hundred.html
- *Riscos das escolas de Java*
  http://www.joelonsoftware.com/articles/ThePerilsofJavaSchools.html
- *Comparação de Sintaxe*
  http://merd.sourceforge.net/pixel/language-study/syntax-across-languages/

Papel de LPs no PDS

- O objetivo de LPs é tornar mais efetivo o processo de desenvolvimento de software (PDS)
- PDS visa geração e manutenção de software de modo produtivo e garantia de padrões de qualidade

Papel de LPs no PDS

- Principais Propriedades Desejadas em um Software
  - Confiabilidade
  - Manutenibilidade
  - Eficiência

Papel de LPs no PDS

- Etapas do PDS
  - Especificação de Requisitos
  - Projeto do Software
  - Implementação
  - Validação
  - Manutenção

História e Origem de LPs e Hw

- 1951-55
  - **Hardware:** computadores a válvula e memórias com linhas de retardo de mercúrio
  - **Métodos:** Linguagens de máquina, primórdios do uso de sub-rotinas e estruturas de dados.
  - **PLs:** uso experimental de compiladores de expressões.

História e Origem de LPs e Hw

- 1956-60
  - **Hardware:** armazenamento em fitas magnéticas, memórias de núcleo de ferrite, circuitos transistorizados.
  - **Métodos:** início do uso de compiladores, gramática BNF, otimização de código, interpretadores, processamento de listas, e métodos dinâmicos de armazenamento.
  - **PLs:**
    - FORTRAN (1957, Aplicações Numéricas),
    - ALGOL 58,
    - ALGOL 60 (programação estruturada),
    - COBOL (programação Comercial)
    - LISP (1959, Programação Funcional).

História e Origem de LPs e Hw

- 1961-65
  - **Hardware:** famílias de arquiteturas compatíveis, discos magnéticos
  - **Métodos:** SO multi-programados, compiladores direcionados à sintaxe.
  - **PLs:**
    - COBOL61,
    - novo ALGOL 60,
    - SNOBOL,
    - JOVIAL, notação APL.

História e Origem de LPs e Hw

- 1966-70
  - Hardware: crescimento explosivo de capacidade e velocidade e decréscimo de custo, minicomputadores, micro-programação, e circuitos integrados.
  - Métodos: sistemas interativo de tempo compartilhado, compiladores com otimização, sistemas de escrita de tradução.
  - PLs:
    - BASIC (1964, Ensino para leigos)
    - APL, FORTRAN 66, COBOL 65, ALGOL 68, SNOBOL 4, PL/I, SIMULA 67, ALGOL-W.

História e Origem de LPs e Hw

- 1971-75
  - **Hardware:** primeiros microprocessadores e microcomputadores, auge dos minicomputadores, sistema de armazenamento de massa, memórias de semicondutores.
  - **Métodos:** Verificação de programas, programação estruturada, início da engenharia de software como disciplina.
  - **PLs:**
    - Pascal (1971, Ensino de estruturada e Simplicidade),
    - *C* (1972, Implementação Unix),
    - Prolog (1972, Programação Lógica)
    - Scheme, COBOL 74, PL/I padrão.

História e Origem de LPs e Hw

- 1976-80
  - Hardware: microcomputadores comerciais, grandes sistemas de armazenamento de massa, programação distribuída.
  - Métodos: tipos abstratos de dados, semânticas formais, programação de tempo real e sistemas embarcados.
  - PLs:
    - SMALLTALK (1972, programação orientada a objetos)
    - Ada (1983, Programaçào concorrente)
    - FORTRAN 77, ML.

História e Origem de LPs e Hw

- 1981-85
  - **Hardware:** computadores pessoais, primeiras estações de trabalho, vídeogames, redes locais, Arpanet.
  - **Métodos:** Programação orientada à objetos, sistemas interativos de programação, editores baseados em sintaxe.
  - **PLs:** Turbo Pascal, Smalltalk-80, uso de Prolog, Ada 83, Postscript.

História e Origem de LPs e Hw

- 1986-90
  - **Hardware:** idade do microcomputador, estações de trabalho de preço médio, arquitetura RISC, comunicação global de dados, Internet.
  - **Métodos:** computação cliente-servidor (e-mail, ftp, terminais remotos)
  - **PLs:**
    - FORTRAN 90,
    - C++ (1985, disseminação da programação OO),
    - SML (Standard ML).

História e Origem de LPs e Hw

- 1991-95
  - **Hardware:** estações de trabalho rápidas e baratas, arquiteturas maciçamente paralelas, imagem, voz e fax sobre a Internet.
  - **Métodos:** sistemas abertos, ambientes de desenvolvimento, infraestruturas nacionais de comunicação de dados, surgimento da WWW.
  - **PLs:** Ada95, linguagens de processos (TCL, PERL), HTML.

História e Origem de LPs e Hw

- 1996-00
  - **Hardware:** popularização de computadores e estação de trabalho, surgimento dos "equipamentos de informação" (celulares e PDAs), vídeo sobre a Internet.
  - **Métodos:** infraestruturas mundiais de comunicação de dados, sistemas Web, arquiteturas 3-tiers e n-tiers, máquinas virtuais independentes deplataforma.
  - **PLs:** Java (1995, mais simples e confiável que C++)
    - Internet, XML, etc ... .

História e Origem de LPs e Hw

- 2000-16
  - **Hardware:** sistemas de informação domésticos, computadores de baixíssimo custo, equipamentos de informação convergentes TV digital e celulares; disseminação de redes wireless .Compactação do hardware em dispositivos poderosos, mas que ainda sofrem com problemas de desempenho vs economia de energia.

História e Origem de LPs e Hw

- 2000-16
  - **Métodos:** multicasting a nível pessoal (Móvel), identidade digital, sistemas Web semânticos, sistemas de auxílio à GC, sistemas convergentes. Sistemas distribuídos, Multicore, virtualização, chamads remotas.
  - **PLs: .NET, JEE, XML, PHP, Ruby, Scala, Python...**

Introdução

- O que é uma linguagem?

Introdução

- O que é uma linguagem?
  - – "Conjunto de regras que estabelecem normas de comunicação".
  - – Caso as partes envolvidas na comunicação falem línguas diferentes, surge a necessidade de um tradutor (intermediário).

Introdução

- O que é uma linguagem de programação?

Introdução

- O que é uma linguagem de programação?
  - Também é um conjunto de regras que estabelecem normas de comunicação entre o programador e o computador.
  - Uma LP deve ser extremamente formal e exata.
  - Uma linguagem ambígua torna-se difícil de ser traduzida para uma linguagem de máquina.

Evolução de LPs

- Dificuldade de Programação em Linguagens de Máquina
- Foco de Primeiras LPs era Eficiência de Processamento e Consumo de Memória
- Baixa Produtividade de Programação
  - Programação Estruturada
  - Tipos Abstratos de Dados
  - Orientação a Objetos

Evolução das LPS

- Primeira Geração:
  - Linguagem de máquina
    - Código de Máquina (0s e 1s).
- Segunda Geração:
  - Linguagem de Montagem - Assembler
- Terceira Geração
  - Imperativas: FORTRAN, Cobol, Basic, Algol, ADA, Pascal, C
  - Lógicas e Funcionais: LISP, ML, Prolog
- Quarta Geração
  - Geradores de Relatórios, Linguagens de Consultas: SQL, CSP
- Quinta Geração
  - LOO : Smalltalk, Java, Eiffel, Simula 67
- Sexta Geração ?
  - Web e Linguagens Dinâmicas : Python, JavaScript, Ruby

Propriedades Desejáveis em LPs

- Legibilidade
  - Marcadores de Blocos

```
if (x>1)
    if (x==2)
      x=3;
else
    x=4;
```

  - Desvios Incondicionais

```
goto
```

  - Duplicação de Significado de Vocábulos

```
this (em JAVA)
*p = (*p)*q;
```

Propriedades Desejáveis em LPs

- Efeitos Colaterais

```
int x = 1;
int retornaCinco() {
  x = x + 3;
  return 5;
}
void main() {
   int y;
   y = retornaCinco();
   y = y + x;
}
```

Propriedades Desejáveis em LPs

- Redigibilidade
  - Tipos de Dados Limitados (FORTRAN)
  - Ausência de Tratamento de Exceções
  - Conflito Ocasional com Legibilidade

```
void f(char *q, char *p) {
   for (;*q=*p; q++,p++);
}
```

Propriedades Desejáveis em LPs

- Confiabilidade
  - Declaração de Tipos

```
boolean u = true;
int v = 0;
while (u && v < 9) {
  v = u + 2;
  if  (v == 6) u = false;
}
```

  - Tratamento de Exceções

```
try {
  System.out.println(a[i]);
} catch (IndexOutofBoundsException) {
  System.out.println("Erro de Indexação");
}
```

Propriedades Desejáveis em LPs

- Eficiência
  - Verificação Dinâmica de Tipos
- Facilidade de Aprendizado
  - Excesso de Características é Prejudicial

```
c = c + 1;
c+=1;
c++;
++c;
```

- Modificabilidade

```
const float pi = 3.14;
```

Propriedades Desejáveis em LPs

- Reusabilidade

```
void troca (int *x, int *y) {
    int z = *x;
    *x = *y;
    *y = z;
}
```

- Portabilidade
  - Rigor no Projeto
  - Pode Contrastar com Eficiência

Especificação de LPs

- Léxico x Sintaxe x Semântica

  `a = b;`

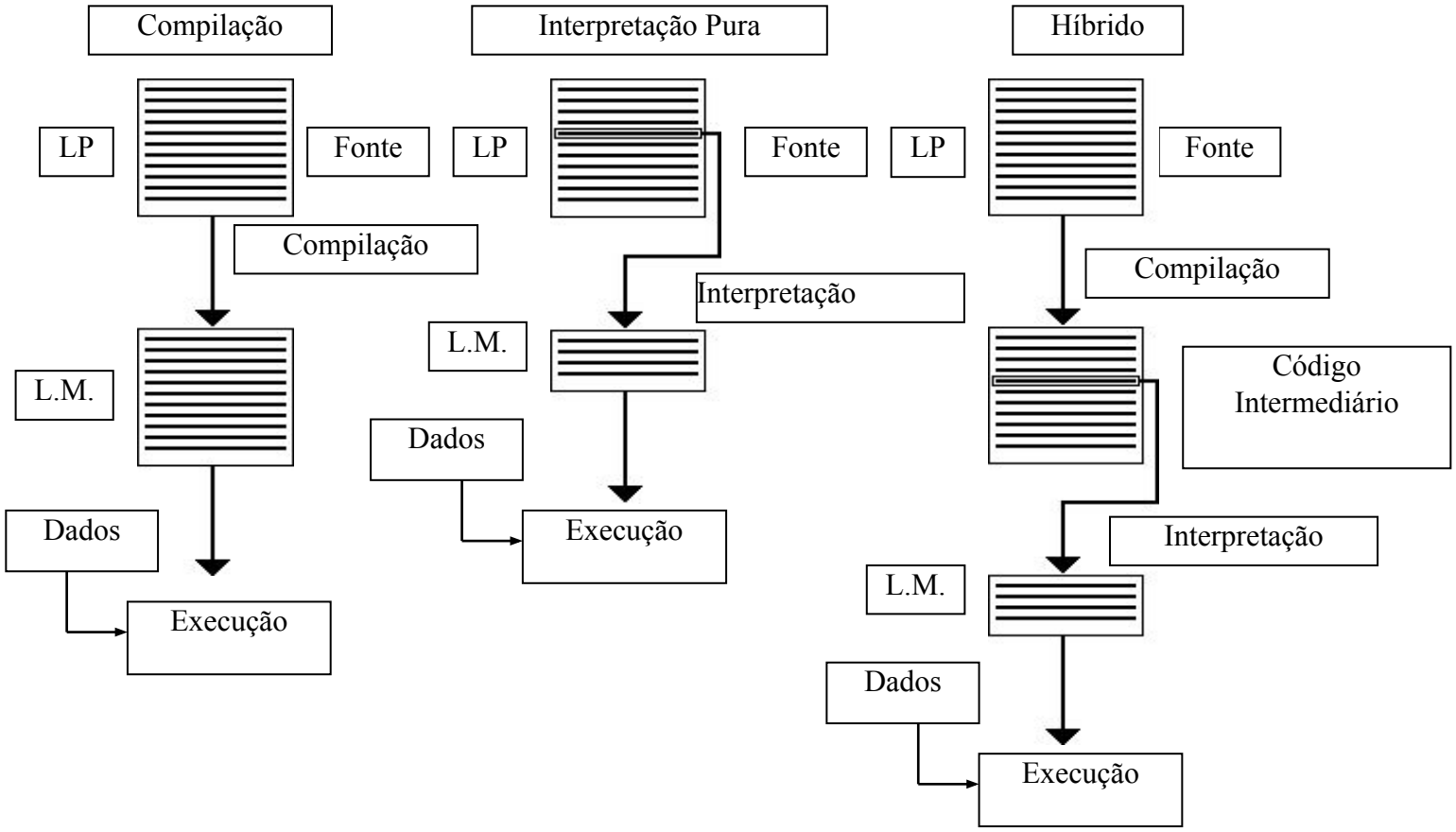
- Sintaxe

  <expressão>::=
  <valor>|<valor><operador><expressão>
  <valor> ::= <número>|<sinal><número>
  <número> ::= <semsinal>|<semsinal>.<semsinal>
  <semsinal> ::= <dígito>|<dígito><semsinal>
  <dígito> ::= 0|1|2|3|4|5|6|7|8|9
  <sinal> ::= + | -
  *<operador> ::= +|-|/|**

Especificação de LPs

- Semântica
  - Enfoque Operacional
- Necessidade de Padronização
  - ISO, IEEE, ANSI, NIST

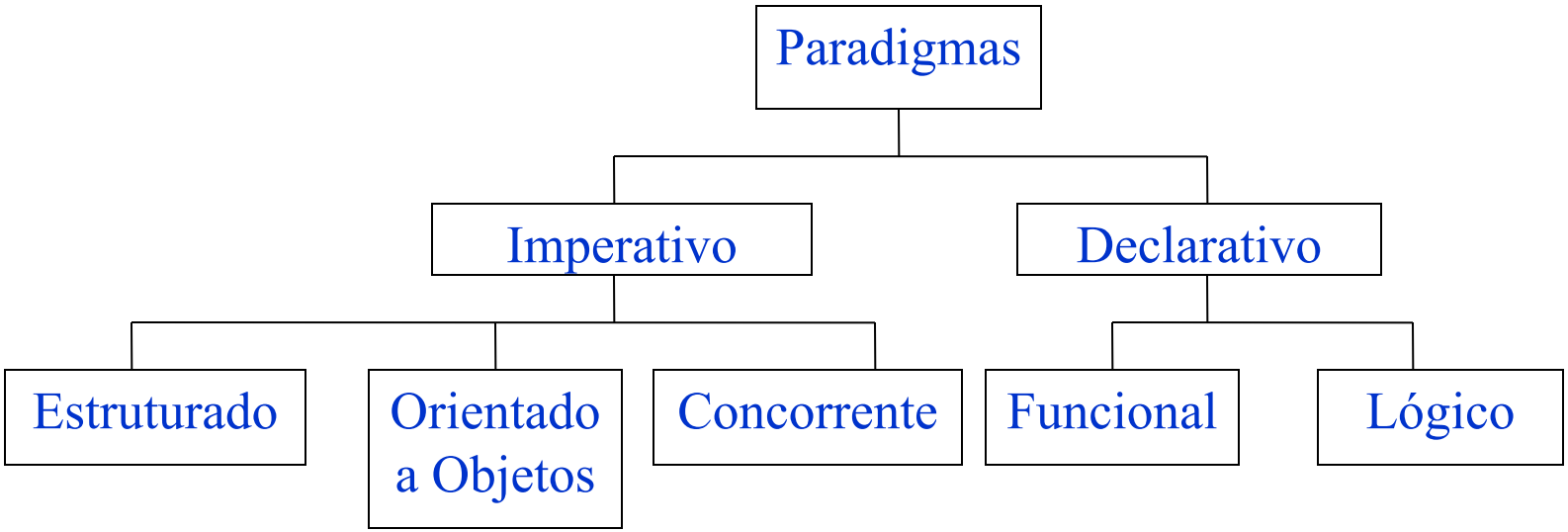
Implementação de LPs



Implementação de LPs

- Compilação
  - Eficiência
  - Problemas com Portabilidade e Depuração
- Interpretação Pura
  - Flexibilidade, Portabilidade e Facilidade para Prototipação e Depuração
  - Problemas com Eficiência e Maior Consumo de Memória
  - Raramente Usada
- Híbrido
  - Une Vantagens dos Outros Métodos
  - JVM

Paradigmas de LPs



Paradigmas de LPs

- Imperativo
  - Processo de Mudanças de Estados
  - Variável, Valor e Atribuição
  - Células de Memória
- Estruturado
  - Refinamentos Sucessivos
  - Blocos Aninhados de Comandos
  - Desestímulo ao uso de desvio incondicional

Paradigmas de LPs

- Orientado a Objetos
  - Abstração de Dados
- Concorrente
  - Processos Executam Simultaneamente e Concorrem por Recursos

Paradigmas de LPs

- Declarativo
  - Especificações sobre a Tarefa a Ser Realizada
  - Abstrai-se de Como o Computador é Implementado
- Funcional
  - Programa Composto por Funções
- Lógico
  - Predicados
  - Dedução Automática