Linguagens de Programação
Conceitos e Técnicas

Introdução

Razões para Estudar 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

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

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

Objetivos de Projeto [T&N]

- Simplicidade e legibilidade:
  - Fácil de escrever, ler, aprender e ensinar;
- Clareza nas ligações:
  - Ligações = amarrações (v. Cap. 2);
- Confiabilidade:
  - Tratamento de exceções, restringir vazamento de memória, tipagem forte, sintaxe/semântica bem definidas, V&V;

Objetivos de Projeto [T&N]

- Suporte:
  - Compiladores acessíveis (baratos, muitas plataformas), cursos, livros, comunidade;
- Abstração:
  - Não ter que reinventar a roda;
- Ortogonalidade;
  - Menor nº de regras excepcionais possível;
  - Cidadãos de primeira classe;

Objetivos de Projeto [T&N]

- Implementação eficiente:
  - Implementações iniciais de Java, Ada e Algol sofreram críticas.

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> ::= +|-|/|**

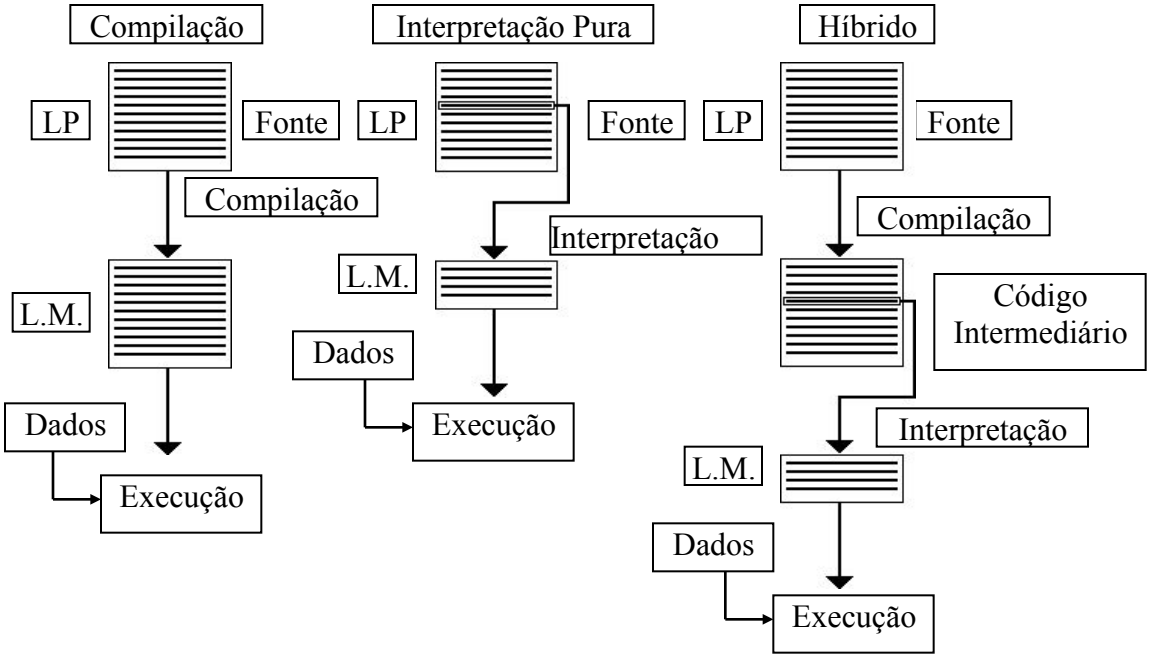
Especificação de LPs

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

Exemplos: ANSI/ISO Cobol 2002, ISO Fortran 2004, ISO Haskell 1998, ISO Prolog 2000, ANSI/ ISO C 1999, ANSI/ISO C++ 2003, ANSI/ISO Ada (2005), ANSI Smalltalk 2002, ISO Pascal 1990.

Outro exemplo: www.jcp.org

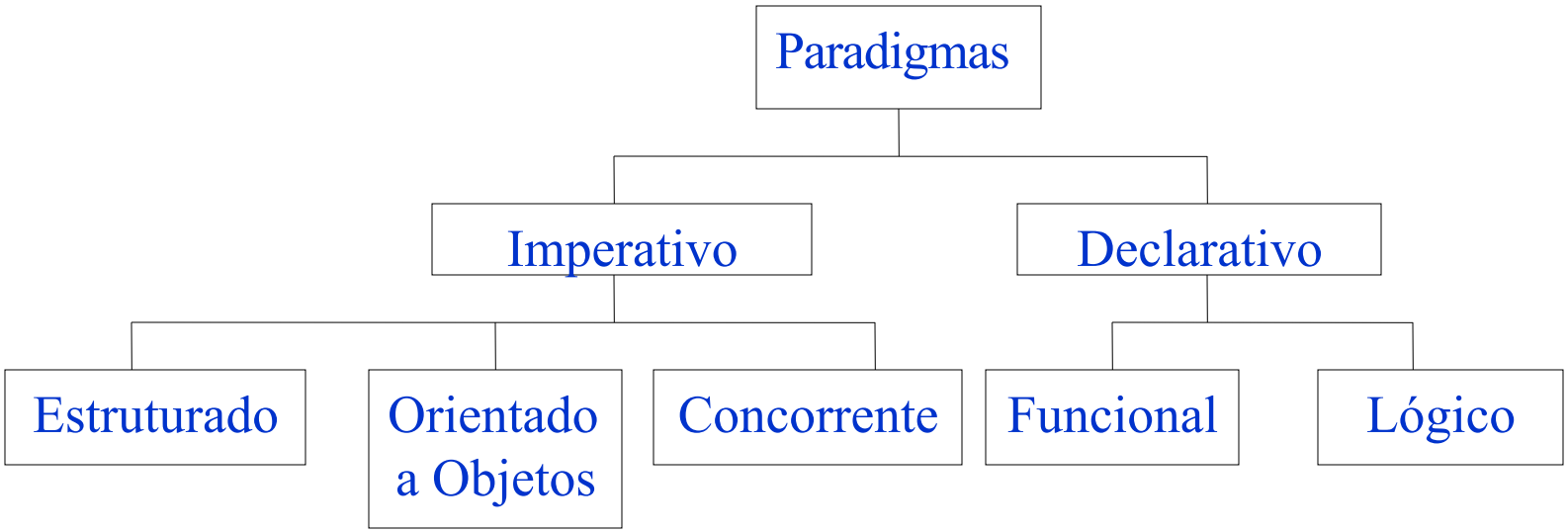
Implementação de LPs



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

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

Origem de LPs

- FORTRAN (1957)
  - aplicações numéricas
- LISP (1959)
  - programação funcional
- ALGOL (1960)
  - programação estruturada
- COBOL (1960)
  - aplicações comerciais

Origem de LPs

- BASIC (1964)
  - ensino para leigos
- PASCAL (1971)
  - ensino de programação estruturada
  - simplicidade
- C (1972)
  - implementação de UNIX
- PROLOG (1972)
  - programação lógica

Origem de LPs

- SMALLTALK (1972)
  - programação orientada a objetos
- ADA (1983)
  - programação concorrente
- C++ (1985)
  - disseminação da programação orientada a objetos
- JAVA (1995)
  - mais simples e confiável que C++
  - Internet

[Tucker & Noonan]