Introdução a Computação

Visão Geral de Linguagens de Programação

O que caracteriza uma Linguagem de Programação?

- Gramática e significado bem definidos
- Implementável (executável) com eficiência ‘‘aceitável’’
- Universal: deve ser possível expressar todo problema computável
- Natural para expressar problemas (em um certo domínio de aplicação)

Aspectos do estudo de linguagens

- Léxico: palavras
- Sintaxe: gramática (forma)
- Semântica: significado
- Pragmática (ex.: metodologias)
- Processadores:
  - compiladores,
  - interpretadores,
  - editores,
  - ambientes visuais ...

Por que tantas linguagens?

- Propósitos diferentes
- Avanços tecnológicos
- Interesses comercias
- Cultura e background científico

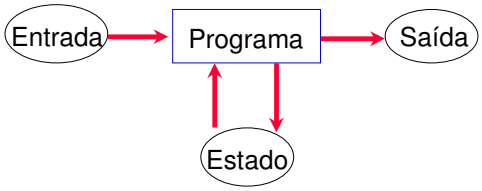
O que é um paradigma de programação?

- Modelo, padrão ou estilo de programação suportado por linguagens que agrupam certas características comuns
- A classificação de linguagens em paradigmas é uma conseqüência de decisões de projeto que impactam radicalmente a forma na qual uma aplicação real é modelada do ponto de vista computacional

O Paradigma Imperativo

- Programas centrados no conceito de um estado (modelado por variáveis) e ações (comandos) que manipulam o estado
- Paradigma também denominado de procedural, por incluir subrotinas ou procedimentos como mecanismo de estruturação
- Primeiro paradigma a surgir e ainda é o dominante

Modelo Computacional do Paradigma Imperativo



Vantagens do modelo imperativo

- Eficiência (embute modelo de Von Neumann)
- Modelagem "natural" de aplicações do mundo real
- Paradigma dominante e bem estabelecido

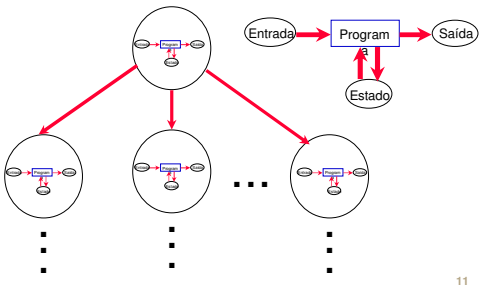
Desvantagens do paradigma imperativo

- Relacionamento indireto entre E/S resulta em:
  - difícil legibilidade
  - erros introduzidos durante manutenção
  - descrições demasiadamente operacionais focalizam o **como** e não **o que**

O Paradigma Orientado a Objetos

- Não é um paradigma no sentido estrito: é uma subclassificacão do imperativo
- A diferença é mais de metodologia quanto à concepção e modelagem do sistema
- A grosso modo, uma aplicação é estruturada em módulos (classes) que agrupam um estado (atributos) e operações (métodos) sobre este
- Classes podem ser estendidas e/ou usadas como tipos (cujos elementos são objetos)

Modelo Computacional do Paradigma Orientado a Objetos



Vantagens do Paradigma Orientado a objetos

- Além de todas as do estilo imperativo
- Classes estimulam projeto centrado em dados:
  - Modularidade
  - Reusabilidade
  - Extensibilidade
- Aceitação comercial crescente - Java

Problemas do Paradigma OO

- Semelhantes aos do paradigma imperativo, mas amenizadas pelas facilidades de estruturação

Tendência: integração de paradigmas

- A principal vantagem é combinar facilidades de mais de um paradigma, aumentando o domínio de aplicação da linguagem
- Exemplos: linguagens lógicas ou funcionais com o conceito de estado e comandos
- A integração deve ser conduzida com muita cautela, para que não se viole os princípios básicos de cada paradigma.

Outras Classificações

- Linguagens de 1a., 2a., 3a. 4a. e 5a. Gerações
  - 1ª: linguagens de Máquina
  - 2ª: linguagens de montagem (Assembly)
  - 3ª: linguagens de Alto Nível (Fortran, Pascal)
  - 4ª: linguagens RAD (Java, Visual, Adabes)
  - 5ª: Linguagens Orientadas a IA (LISP, Prolog)

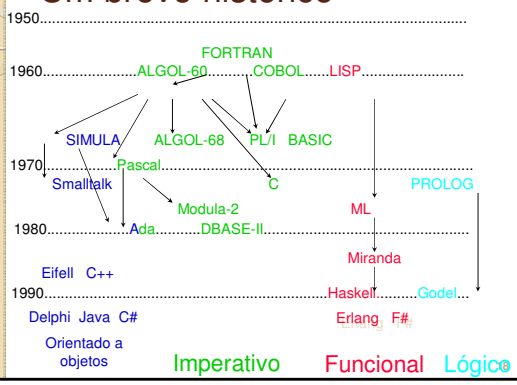
Outras Classificações

- Programação seqüencial versus concorrente versus paralela
  - Sequencial: programação baseada na execução sequencial de comandos
  - Concorrente: programação baseada na execução não sequencial e cooperativa de comandos
  - Paralela: programação baseada na divisão de tarefas independentes

Outras Classificações

- Programação linear versus programação visual (visual programming)
  - Ambientes visuais de desenvolvimento avançado de aplicações
  - Baseado na filosofia de reuso de Componentes ou Paradigma Orientado a Componentes
- ...

Um breve histórico



Evolução centrada em níveis crescentes de abstração

- Linguagens de máquina
  - Endereços físicos e *operation code*
- Linguagens Assembly
  - Mnemônicos e labels simbólicos
- Linguagens de "alto nível"
  - Variáveis e atribuição (versus acesso direto à memória)
  - Estruturas de dados (versus estruturas de armazenamento)

Evolução centrada em níveis crescentes de abstração

- Linguagens de "alto nível"
  - Estruturas de controle (versus *jumps* e *gotos*)
  - Estrutura de blocos como forma de encapsulamento
  - Generalização e parametrização (abstração de tipos de valores)

Compiladores

- Programa
  - A partir de um código-fonte escrito em uma linguagem, cria uma programa equivalente, mas escrito na linguagem objeto
  - Composto por:
    - Analisador Léxico
      - Verifica a compatibilidade com determinado alfabeto
    - Analisador Sintático
      - Determinação da estrutura gramatical de uma sentença
    - Gerador de Código
    - Otimizador

Compilador