Capítulo III - Linguagens de Programação

- **Porque Estudar Ling. de Programação?**
- **Quais são as formas de classificar Linguagens?**
- **Quais são os Conceitos Fundamentais de Linguagens de Programação?**
- **Como Avaliar/Projetar Ling. de Programação?**
  - **Características**
  - **Critérios**
- **Como Especificar Ling. de Programação?**
- **Como Implementar Ling. de Programação?**

III.1 – Motivos para Estudar Ling. de Prog.

- **Aumentar a capacidade de expressar idéias**
  - **Maior conhecimento facilita a expressão**
- **Conhecer melhor para Escolher melhor**
  - **O maior conhecimento leva ao uso mais eficaz dos recursos e facilidades**
  - **Possibilita escolhas mais adequadas**
  - **Permite a evolução da área**
- **Facilitar o aprendizado de novas linguagens**
  - **Evolução contínua exige atualização constante**
  - **Maior conhecimento, facilita/acelera o aprendizado**
- **Entender melhor a implementação**
  - **Permite entender o porquê das coisas**
  - **Permite o uso mais eficiente e racional dos recursos**
- **Poder projetar novas linguagens**
  - **O entendimento do que existe, permite avaliar e propor novas soluções**
  - **Uso dos modelos de especificação e técnicas de compilação na solução de problemas convencionais.**
- **Contribuir para o avanço da computação**
  - **Pq algumas linguagens se sobressaem?**
  - **Pq Linguagens boas podem demorar para serem reconhecidas/aceitas?**
  - **Se quem usa conhece, o tempo para aceitação de boas linguagens será reduzido.**

III.2 – Classificações de Linguagens

Quanto a Geração (evolução)

1ª Geração

- Linguagens de Máquina

2ª Geração

- Linguagens Simbólicas (Assembly)

3ª Geração

- Linguagens orientadas ao usuário
  - Projetadas para profissionais de Informática
- procedimentais (procedurais ou imperativas)
  - instruções: E/S, aritméticas e lógicas, de controle
  - ex.: Fortran, Algol, Pascal, C, Ada
- declarativas
  - Funcionais – teoria das funções recursivas (Lisp)
  - Lógicas – lógica de predicados (Prolog)

4ª Geração

- **Linguagens orientadas a aplicação**
  - Projetadas para usuários finais
  - integradas com BD
  - uso de formulários, menus, gráficos, etc
- **Visam simplificar o desenvolvimento e reduzir o custo de manutenção**
- **Ex.: lotus, excel, sql, L. gráficas e de simulação**

5ª Geração

- **Facilidades para representação do conhecimento**
- **Usadas na área de Inteligência Artificial**

Quanto ao domínio de aplicação

Científicas

- Estruturas de dados simples, pouca E/S, alta utilização de CPU
- Fortran, Algol e suas derivadas

Comerciais

- Facilidades para representação/manipulação de dados alfanuméricos
- Facilidade para geração de relatórios
- Cobol, Planilhas eletrônicas e sistemas de BD

Programação de sistemas

- desenvolvimento de software básico
- recursos de baixo nível e bom desempenho
- PL/I, Algol e C

Linguagens para propósitos especiais

- Linguagens de Scripting: Javascript, perl, php
- Linguagens para IA: Lisp, Prolog
- Linguagens de simulação : GPSS
- Linguagens Tempo Real : RTCC, RTJava
- ...

Quanto a Implementação

Compilação

- geração de código de baixo nível
- vantagem – eficiência na execução
- ex. típicos: Fortran, cobol, C, Ada, Pascal

Interpretação

- simulação de uma máquina em software
- vantagem – rapidez no desenvolvimento
- desvantagem – maior tempo de execução
- exemplos típicos : Basic, APL, PHP, Linguagens de controle, Linguagens de consulta

Implementação Híbrida

- compilação para um código intermediário

  O Código Int. pode ou não ser transparente
- Interpretação do código intermediário
- reúne vantagens e desvantagens dos anteriores
- exemplos : Java, algumas versões de Pascal

Quanto ao Paradigma de Programação

Classificação usual

- L. Imperativas (Procedurais, Estruturadas)
- Linguagens Baseadas/Orientadas a Objetos
- Linguagens Lógicas
- Linguagens Funcionais

Classificação de Peter Wegner

- Imperativas (como fazer!)
  - Estruturadas (blocos)
  - Baseadas/Orientadas a Objetos
  - Distribuídas
- Declarativas (o que fazer!)
  - Funcionais (funções recursivas)
  - Lógicas (cálculo proposicional)
  - Relacionais (álgebra relacional)

Principais aspectos dos diversos Paradigmas

- **Imperativo (Procedural, Estruturado)**

Características básicas

- baseadas na arquitetura de *von Neumann*
- blocos, procedures (funções) e recursão
- geralmente são fortemente tipadas
- implementação baseada em pilha
- estruturas de controle convencionais
- natureza top-down

Vantagens

- Facilita a organização de programas e sistemas
- Permite ocultar informação
- Eficiência
- Algumas também possuem facilidades para programação de grandes sistemas (Módula e ADA)

Desvantagens

- ger. de grandes sistemas pode ser trabalhoso
- abstrações menos naturais
- sintaxe e semântica mais complexas
- reuso possível, porém não explícito

Exemplos

- Algol, Pascal, C, Modula e ADA

Aplicações

- programação científica
- programação de sistemas

- **Baseado/Orientado a Objetos**

Características básicas

- objetos e classes
- atributos, métodos e mensagens
- abstração de dados (ocultamento de informação)
- ligação dinâmica e polimorfismo
- herança

Vantagens

- reutilização
- produtividade
- projeto e programação integrados
- afinidade com o mundo real
- facilita manutenção/extensão de software

Desvantagens

- complexidade conceitual maior
- eficiência pode ficar comprometida
- implementações mal projetadas / programadas

Exemplos

- Simula, Smaltalk (pura)
- C++, Java, Delphi (Object Pascal)
- Dialetos de praticamente todas as linguagens
- Outros paradigmas: Lógicas e Funcionais

Aplicações

- programação de grandes sistemas em geral

• Lógico

Características básicas

- linguagem declarativa
- baseada no cálculo dos predicados
- fatos : pai (João, Carlos); mãe (Maria,Carlos)
- regras: irmãos (x,y) :- mãe (M,x), mãe(M,y),
  pai (P,x), pai(P,y)
- metas (consultas) : irmãos (Carlos, Ana)
- processo de inferência para produzir resultados

Exemplos

- PROLOG e seus dialetos

Vantagens

- facilidades específicas para suas áreas de aplicação
- Preocupação em o que fazer, e não como fazer!

Desvantagens

- problemas de eficiência
- carência de métodos para programação de grandes sistemas

Aplicações

- área de IA – sistemas de conhecimento, sistemas especialistas, proc. de linguagem natural
- gerenciamento de BD
- prova de teoremas

- **Funcional**

  Características básicas

  - baseado em funções matemáticas
  - funções primitivas + formas para construção de funções complexas
  - não possuem variáveis, atribuição e iteração
  - podem ou não ser puras
    - presença de alguns recursos imperativos
  - repetição feita via recursão
  - execução ≡ avaliação de funções
  - normalmente interpretadas, embora possam ser compiladas

  Vantagens

  - sintaxe e semântica mais simples
  - nível de programação mais elevado
  - concorrência controlada pelo sist. de execução

  Desvantagens

  - baixa eficiência
  - comunidade reduzida de usuários

  Exemplos

  - LISP,Scheme, ML, Haskell

  Aplicações

  - Computação simbólica
  - Processamento de listas
  - Aplicações de IA
    - Sistemas especialistas
    - Representação de conhecimento
    - Processamento de linguagem natural

III.3 - Conceitos Fundamentais de Linguagens de Programação

- Constantes e variáveis
- Declarações explícitas X implícitas
- Vinculação estática X dinâmica
- Tipos e Estruturas de Dados
  - Tipos primitivos
  - Tipo string (cadeia de caracteres)
  - Ordinais definidos pelo usuário
    Enumeração, subrange
  - Tipos estruturados
    Array, registro, união, conjunto
  - Ponteiros
- Tipagem forte
  - Todos os erros de tipo são detectados
    Tempo de compilação ou de execução
- Verificação de tipos: estática X dinâmica
- Escopo (regras de visibilidade)
- Abstração de Dados (tipo de dados abstratos)
  - Encapsulamento envolvendo dados e operações
  - Usados como tipos em declarações

- **Expressões e Atribuição**
  - Aritméticas, relacionais e lógicas (booleanas)
  - Operandos e operadores
  - Associatividade
  - Sobrecarga de operadores
  - Compatibilidade e conversão de tipo

- **Estruturas de Controle**
  - Comandos compostos
  - Seleção : if, case/switch
  - Repetição : while, repeat, for
  - Desvios incondicionais
  - Entrada / Saída

Abstração Funcional

- subprogramas (funções e procedimentos)
  - princípios básicos
    - definição (interface), corpo e chamada
  - parâmetros (formais x reais)
    - número, ordem e tipo
    - mecanismos de passagem
      - valor, referência, nome
    - procedimentos podem ser parâmetros?
  - alocação – estática x dinâmica
  - variáveis locais X variáveis globais
  - sobrecarga, genericidade
  - recursão, co-rotinas e interfaces

- formas de agrupamento
  - units, packages, classes/objetos, módulos

- Programação de grandes sistemas
  - compilação em separado (ou independente)

- **Concorrência**
  - **Diferentes níveis**
    - **Instrução, comando, unidade, programa**
  - **Questão de linguagem : comando e unidade**
  - **Física X lógica**
  - **Tarefas e threads**
  - **Sincronização**
    - **Cooperação e Competição**
  - **Mecanismos de sincronização**
    - **Monitores**
    - **Semáforos**
    - **Troca de mensagens**

- **Exceções**
  - **Evento (errôneo ou não) detectável por software ou hardware que exige processamento especial**
  - **Exemplos**
    - **Leitura de disco**
    - **overflow de ponto-flutuante**
    - **Índice inválido**
  - **tratada na mesma ou em outra unidade**
  - **Vantagens**
    - **Legibilidade**
    - **Confiabilidade**
    - **Custo e tamanho de programas (propagação)**
  - **Diferentes linguagens – diferentes formas de implementação**
    - **PL/I – On <condição>**
    - **ADA – exception when <condição> => <ação>**
    - **C++ - try – throw - catch**
    - **Java - try – throw – catch - finally**

III.4 – Critérios/Características a serem observados no projeto/avaliação de Linguagens de Programação

- A LP deve ser adequada às aplicações alvo
  - possuir recursos necessários e suficientes
  - o programador deve ter domínio da Linguagem
- Correção e confiabilidade
  - características negativas
    goto, variáveis globais,
    sinônimos (par. por referência, ponteiros)
  - características positivas
    tipagem forte, exceções, pré e pós-condições
- Eficiência
  - compilação e execução
- Portabilidade
- Extensibilidade
- Boa definição (formal se possível!!!)
  - Legibilidade (usabilidade)
    - fácil de ler, escrever e entender programas
    - simplicidade e clareza sintática
    - clareza e concisão semântica
  - Uniformidade / ortogonalidade
  - Manutenibilidade (facilidade de manutenção)

- **Provabilidade**
  - **o programa representa a intenção do programador?**
  - **o PO é equivalente ao PF?**
  - **a máquina executa o PO corretamente?**
- **Custo**
  - **Função de várias características**
    - **facilidade para desenvolvimento de programas**
    - **compilação / execução**
    - **manutenção**
    - **confiabilidade**
- **Ambiente de programação**
  - **Facilidades para edição / testes / depuração**
  - **Suporte operacional – bibliotecas nativas**
  - **Facilidades para compilação em separado**
  - **Interface gráfica**

Critérios e características importantes no projeto e avaliação de linguagens de programação

(Sebesta 2004)

Características	Critérios		
	Legibili-dade	Capacidade de escrita	Confiabili-dade
Simplicidade/ortogonalidade	X	X	X
Estruturas de controle	X	X	X
Tipos de dados e estruturas	X	X	X
Boa definição sintática	X	X	X
Suporte para abstração		X	X
Expressividade		X	X
Verificação de tipos			X
Manipulação de exceções			X
Aliasing restrito			X

Influências no Projeto de Linguagens

- Arquitetura do computador
  - (Von Neumann – variáveis/memória)
- Metodologias de Programação
  - projeto top-down
    decomposição funcional
  - orientação a objetos
    herança, ligação dinâmica
  - programação orientada a processos
    concorrência
- Linguagens predecessoras
  - O que valea a pena ser preservado?

Trade-offs (compromissos) no projeto de L. P.

- critérios/características conflitantes
- conciliação desses critérios / características
  - tarefa do projetista

    confiabilidade X custo de execução
    expressividade X legibilidade
    flexibilidade X segurança
    ortogonalidade X simplicidade

III.5 – Como especificar Linguagens de Programação

- Especificação (Formal, Informal ou Mista) dos aspectos:
  - Léxicos, Sintáticos e Semânticos
- Diferentes graus e modelos de Formalização

Aspectos	Mecanismos Formais
Léxicos	GR, AF e ER
Sintáticos	GLC, Autômatos de Pilha
Semânticos / tradução	Gramáticas de Atributos Semântica Denotacional Semântica de Ações

- Aspectos Semânticos e de tradução
  - Uso de Esquemas Mistos

GLC + AÇÕES SEMÂNTICAS

↓

Tradução Dirigida pela Sintaxe

↓

Geração de Código

Limite entre Especificação Léxica e Sintática

- O que é Léxico?
- O que é Sintático?

- **Porque especificar separadamente?**
  - Aspectos léxicos são mais simples
  - Aumento da legibilidade da Esp. Sintática
  - Maior eficiência na Análise

Limite entre Esp. Sintática e Semântica

- O que é Sintático?
- O que é Semântico?

Divisão não muito clara:

1. Sintaxe independente de contexto
2. Sintaxe dependente de contexto
3. Significado Semântico

Na prática: "1." é considerado sintático
"2." e "3." são considerados semânticos

Determinados aspectos podem ser considerados tanto sintáticos quanto semânticos :

- Constantes usadas na definição de intervalos
- Uso restrito de tipos em determinados contextos
- Dimensões de um array

III.6 – Implementação de Linguagens de Programação

♦Formas de Implementação

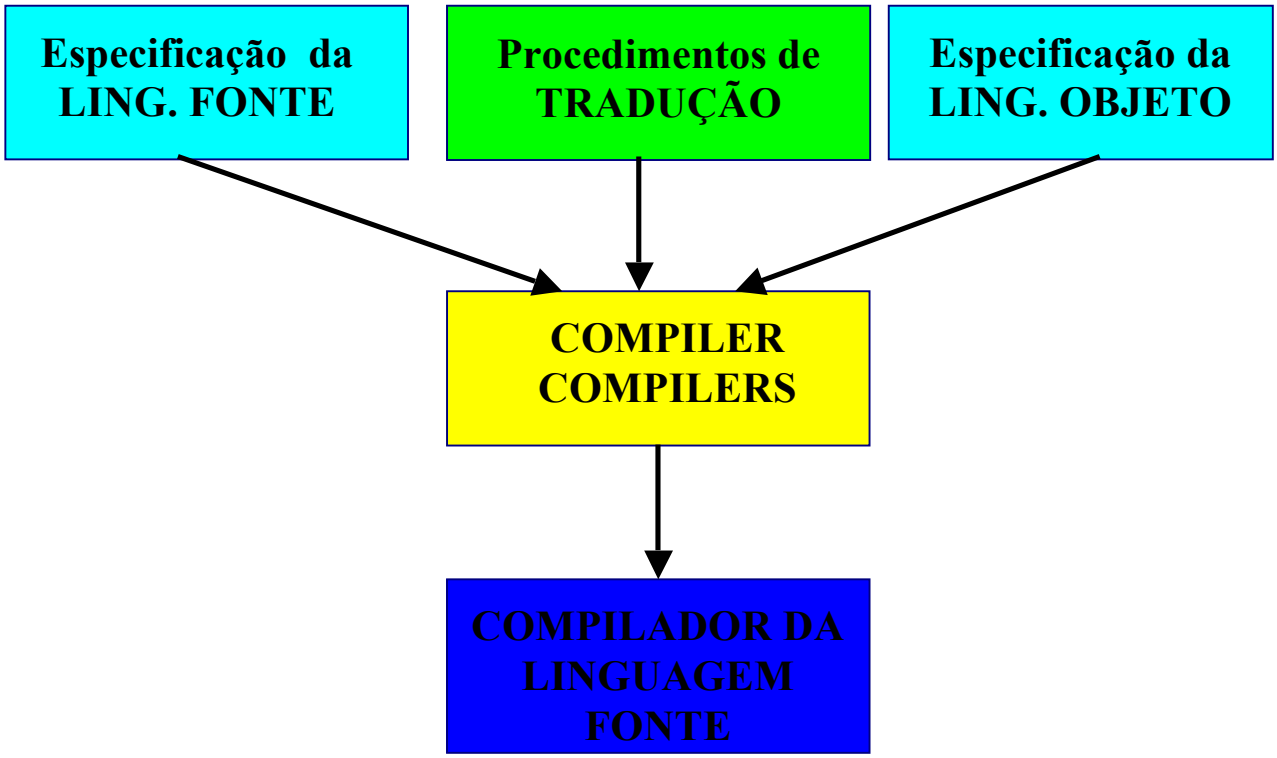
- ♦ Tradução (compilação, pré-processamento, etc...)
- ♦ Interpretação
- ♦ Implementações híbridas (compilação + interpretação)

♦Ferramentas usadas na implementação de Compiladores

Além de ambientes/softwares convencionais, são usadas ferramentas específicas, tais como:

- ♦ Editores de Gramáticas e Autômatos
  - ♦ Gráficos e Textuais
- ♦ Geradores de Analisadores Léxicos
  - ♦ LEX, GALEX, GALS e Gal
- ♦ Geradores de Analisadores Sintáticos
  - ♦ YACC, LADE, SYNTAX, GAS e GALS
- ♦Avaliadores de atributos, Otimizadores e geradores de código

♦ Compiler - Compilers



Aspecto comum : Ferramentas baseadas (parcial ou totalmente) em mecanismos formais,

Trabalho sobre Linguagens de Programação

1 – Ler as anotações de aula deste Capítulo III

2 – Complementar o estudo dos tópicos aqui apresentados, lendo o capítulo sobre avaliação de LP dos seguintes livros da bibliografia (cópia no xerox do ctc):

1. **WATT, D., Programming Language Design Concepts, John Wiley and Sons, Ltd, 2004.**
2. **SEBESTA, R.W., Conceitos de Linguagens de Programação, ed. Bookman, 6. edição, 2005.**
3. **APPLEBY, D., Programming Languages – Paradigm and Practice. Ed McGraw-Hill, Inc., 1991.**
4. **www**

3 – Escolher uma Linguagem de programação, descrevê-la e fazer uma avaliação da mesma, levando em conta os critérios e as características tipicamente utilizados na avaliação de linguagens (incluindo, obrigatoriamente, os apresentados na seção 4 deste capítulo).

Atenção: A avaliação elaborada deverá ser bem fundamentada e ilustrada com exemplos que representem as características que contribuem positiva ou negativamente para que os diferentes critérios sejam bem/mal avaliados.

Observações:

1 – Entregar relatório (impresso) contendo: introdução, descrição da LP escolhida; Avaliação elaborada; conclusão e bibliografia.

2 – A descrição da linguagem escolhida deverá ser sintética (priorizando aspectos gerais em detrimento de detalhes sintáticos) e deverá abordar, entre outros, os seguintes aspectos:

- **2.1 – Breve histórico da Linguagem;**
- **2.2 – Classificação (seção 2);**
- **2.3 – Conceitos fundamentais (seção 3)**

4 - O Trabalho deverá ser feito em duplas e o Relatório deverá ser entregue até 07 de outubro de 2013.