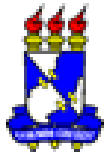
Introdução ao Estudo de Linguagens de Programação

Prof. Alberto Costa Neto
alberto@ufs.br

Linguagens de Programação



Departamento de Computação
Universidade Federal de Sergipe

Conteúdo

- Razões para estudar LP
- Domínios de programação
- Critérios de avaliação de linguagens
- Influências sobre o projeto de linguagens
- Paradigmas de LP
- Trade-offs no projeto de linguagens
- Métodos de implementação
- Ambiente de Programação

Razões para Estudar LP

- Maior capacidade de expressar idéias
  - Aprendizado de novas construções de LP
    - Mesmo que a LP não possua a capacidade aprendida, pode ser simulada (Objetos)
  - Limitações de uma linguagem
    - Tipos de estrutura de dados
    - Tipos de estrutura de controle
    - Tipos de abstrações que podem ser utilizadas
  - Formas de algoritmos também são limitadas

Razões para Estudar LP

- Maior conhecimento para escolha de LP apropriadas
  - Não conhecimento dos conceitos de LP
    - Utilização da LP familiar mesmo que não seja adequada ao novo projeto
  - Conhecimento dos recursos das LPs permite uma escolha mais consciente

Razões para Estudar LP

- Maior habilidade para aprender novas LP
  - Compreensão dos conceitos fundamentais
    - Facilita a percepção de como estes são incorporados ao projeto da LP aprendida
    - Facilita a leitura e o entendimento de manuais e do "marketing" relacionado a linguagens e compiladores

Razões para Estudar LP

- Entender melhor o significado da implementação
  - Visualização de como o computador executa várias construções da linguagem
    - Entendimento da eficiência relativa de construções alternativas a serem escolhidas
  - Capacidade de utilizar uma linguagem de forma mais inteligente, como ela foi projetada
    - String x StringBuffer em Java
  - Facilita a detecção e correção de certos *bugs*
    - Comparação de Strings com == em Java

Razões para Estudar LP

- Aumento da capacidade de projetar novas linguagens
  - Um exame crítico das linguagens de programação ajuda
    - No projeto de sistemas complexos
    - A examinar e avaliar esses produtos

Razões para Estudar LP

- Avanço global da computação
  - Em alguns casos uma linguagem não se torna popular porque aqueles com capacidade de optar ainda não estavam familiarizados com os conceitos de LPs
  - Permanência no FORTRAN em detrimento do ALGOL 60 (mais elegante, melhores estruturas de controle, recursão e bloco) na década de 60

Domínios de Programação

- Computadores são utilizados em áreas bem diferentes
  - Controle de usinas elétricas nucleares
  - Armazenamento e processamento de informações bancárias
  - Medicina
  - Computação pessoal
- Têm sido desenvolvidas LP para diversos domínios

Domínios de Programação

- Aplicações científicas
  - Estrutura de dados simples
  - Computações aritméticas com ponto flutuante
  - Estrutura de dados comuns: array e matriz
  - Estruturas de controle comuns: laços de contagem e de seleções
  - FORTRAN
  - ALGOL 60 e suas descendentes

Domínios de Programação

- Aplicações comerciais
  - COBOL
  - Produz relatórios elaborados
  - Maneira precisa de descrever e armazenar números decimais e dados caracteres
  - Capacidade de especificar operações aritméticas decimais

Domínios de Programação

- Inteligência Artificial
  - Uso de computações simbólicas em vez de numéricas
  - Programação funcional (LISP)
  - Programação lógica (Prolog)

Domínios de Programação

- Programação de sistemas
  - Sistemas operacionais e ferramentas de suporte
  - Décadas de 60 e 70 alguns fabricantes de computadores desenvolveram LP de alto nível
    - IBM – linguagem PL/S
    - Digital – BLISS
    - Burroughs – Extended ALGOL
  - Unix desenvolvido quase inteiramente em C

Domínios de Programação

- Software Web
  - Markup Languages (XHTML)
    - Não são LPs
    - JSP Standard Tab Library (JSLT)
    - Extensible Stylesheet Language Transformations (XSTL)
  - Java (Applets + Servlets + JSP...)
  - Linguagens de Scripting
    - JavaScript
    - Perl

Papel de LPs

- Tornar o processo de desenvolvimento de software (PDS) mais efetivo
- Propriedades Desejadas em um Software
  - Confiabilidade, Manutenibilidade, Eficiência
- Etapas do PDS
  - Especificação de Requisitos
  - Projeto do Software
  - Implementação
  - Validação
  - Manutenção

Critérios de Avaliação de Linguagens

- Legibilidade
- Redigibilidade
- Confiabilidade
- Custo

Critérios e Características de LP

	Critério		
	Legibilidade	*Redigibilidade*	*Confiabilidade*
Simplicidade	x	x	x
Ortogonalidade	x	x	x
Tipos de Dados	x	x	x
Sintaxe	x	x	x
Suporte a Abstração		x	x
Expressividade		x	x
Checagem de Tipos			x
Tratamento de Exceções			x
Restrição a Alias			x

Critérios de Avaliação de Linguagens

- Legibilidade
  - Facilidade com que os programas podem ser lidos e entendidos
  - Facilidade de manutenção é, em grande parte, determinada pela legibilidade dos programas

Critérios de Avaliação de Linguagens

- Legibilidade
  - Simplicidade global
    - Pequeno número de componentes básicos
    - Não possuir multiplicidade de recursos
      - Mais de uma maneira de realizar uma operação particular
    - Evitar Sobrecarga de Operador
      - Usar + para soma de vetores

Critérios de Avaliação de Linguagens

- Legibilidade
  - Ortogonalidade
    - Um conjunto relativamente pequeno de construções primitivas pode ser combinado, em um número relativamente pequeno de maneiras, para construir as estrutura de controle e de dados da linguagem
      - Vetor e Registro podem ser passados como parâmetros?
      - E podem ser retornados de funções? Pascal não suporta!
      - Vetor de vetores são permitidos?
    - O significado de um recurso ortogonal de linguagem é livre do contexto de sua ocorrência em um programa

Critérios de Avaliação de Linguagens

- Legibilidade
  - Ortogonalidade
    - Torna a linguagem mais fácil de aprender
    - Diminui as restrições de programação
    - Muita ortogonalidade também pode causar problemas
    - Simplicidade
      - Combinação de um número relativamente pequeno de construções primitivas
      - Uso limitado do conceito de ortogonalidade

Critérios de Avaliação de Linguagens

- Legibilidade
  - Instruções de controle
    - Restrição à instrução de desvio incondicional (goto)
    - Incluir instruções suficientes com o objetivo de eliminar a necessidade de utilização de goto (if, case, for, while, repeat, ...)
    - As instruções de controle devem ser bem projetadas
  - Tipos de dados e estruturas
    - Facilidade para definir tipos e estrutura de dados auxilia a legibilidade
      - Falta de um tipo boolean em C
        - » usar 0 como false e 1 como true

Critérios de Avaliação de Linguagens

- Legibilidade
  - Sintaxe
    - Identificadores (Poucas restrições na escolha)
      - ANSI BASIC (1 Legra + 1 Dígito) – Ex: B2
      - FORTRAN 77 (6 caracteres) – Ex: NUMALU
    - Palavras especiais
      - begin end x enf if + end loop
      - FORTRAN permite usá-las em nomes de variáveis

Critérios de Avaliação de Linguagens

- Legibilidade
  - Sintaxe (continuação)
    - Forma e Significado
      - * em C (*p = (*p)*q;)
        - » *Retorno do endereço apontador por p*
        - » *Retorno do conteúdo da célula apontada por p*
        - » *Operação de multiplicação*
      - this em Java
        - » Referência ao próprio objeto
        - » Chamada a outro construtor da classe

Critérios de Avaliação de Linguagens

- Redigibilidade
  - Facilidade com que uma linguagem pode ser utilizada para criar programas para o domínio de problema escolhido
  - Simplicidade e Ortogonalidade
    - Pequeno número de construções primitivas
    - Um conjunto consistente de regras para combinar as construções da linguagem

Critérios de Avaliação de Linguagens

- Redigibilidade
  - Suporte a abstração
    - Habilidade para definir e usar estruturas e operações complexas de forma que muitos detalhes sejam ignorados
      - Processos (subprogramas)
      - Dados (estruturas de dados)
        - » FORTRAN não suporta registros
  - Expressividade
    - Facilidade de expressar computação em programas pequenos (++ em C e Java)
    - for (em C e Java) permitem criar laços com contadores mais facilmente do que com o while

Critérios de Avaliação de Linguagens

- Confiabilidade
  - Checagem de tipos (estática x dinâmica)
  - Tratamento de exceções (try / catch / finally)
  - Aliasing
    - Existência de diferentes métodos de acesso a uma mesma célula de memória
    - Mais de um apontador para a mesma variável
  - Legibilidade e Redigibilidade
    - Quanto mais fácil descrever e ler, maior a confiabilidade do programa

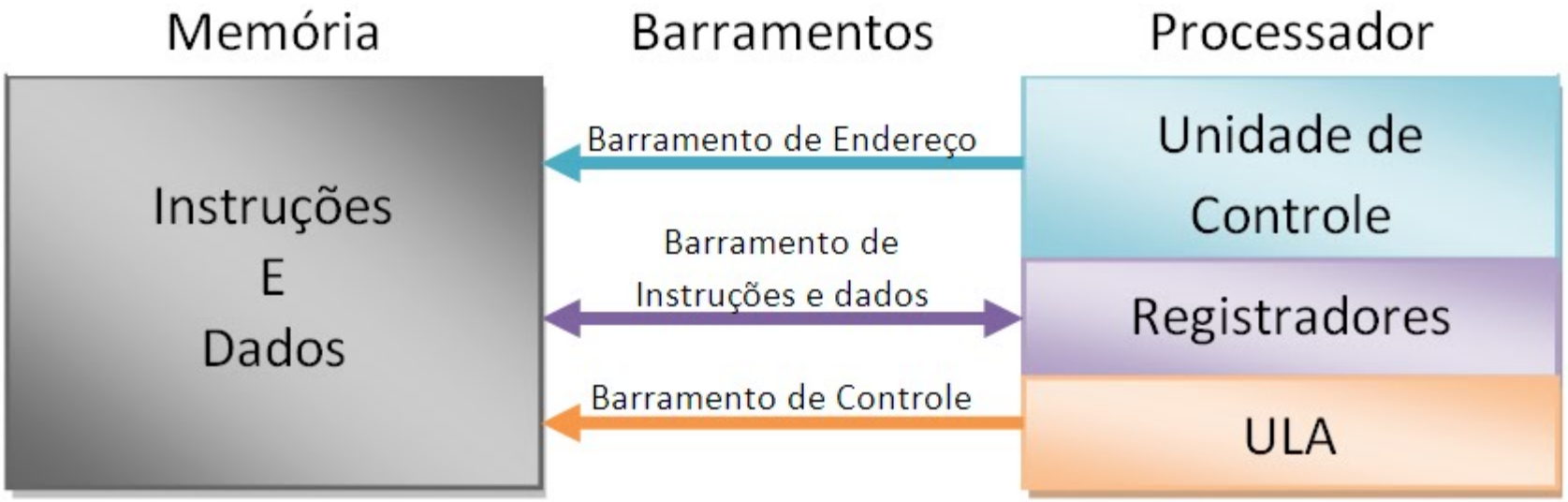
Critérios de Avaliação de Linguagens

- Custo
  - Treinamento de programadores
  - Escrever programas na linguagem
  - Compilar programas na linguagem
  - Executar programas
  - Sistema de implementação da linguagem
  - Custo da má confiabilidade
  - Manutenção dos programas

Outros critérios

- **Eficiência** (checagem de tipos e exceções)
- **Modificabilidade** (interfaces x implementação)
- **Portabilidade**
  - Programas podem rodar em diferentes ambientes/plataformas
  - Normalmente perde em eficiência de execução
- **Facilidade de Aprendizado**
- **Generalidade**
  - Poder ser aplicada em uma ampla gama de situações
- ***Well-definedness***
  - Definição completa e precisa da linguagem

Influências sobre o projeto de linguagens



- Arquitetura do computador (vou Neumann)
- Metodologias de programação
  - Orientada para o processo
  - Orientada a dados

Especificação de LPs

- Necessário definir a **forma** de se escrever programas válidos e **como** estes devem se comportar
- **Léxico x Sintaxe x Semântica**

C `a = b;`

- **Léxico:** **“a” “b” “=” “;”** fazem parte do vocabulário
- **Sintaxe:** **Sentença designa um comando válido**

<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

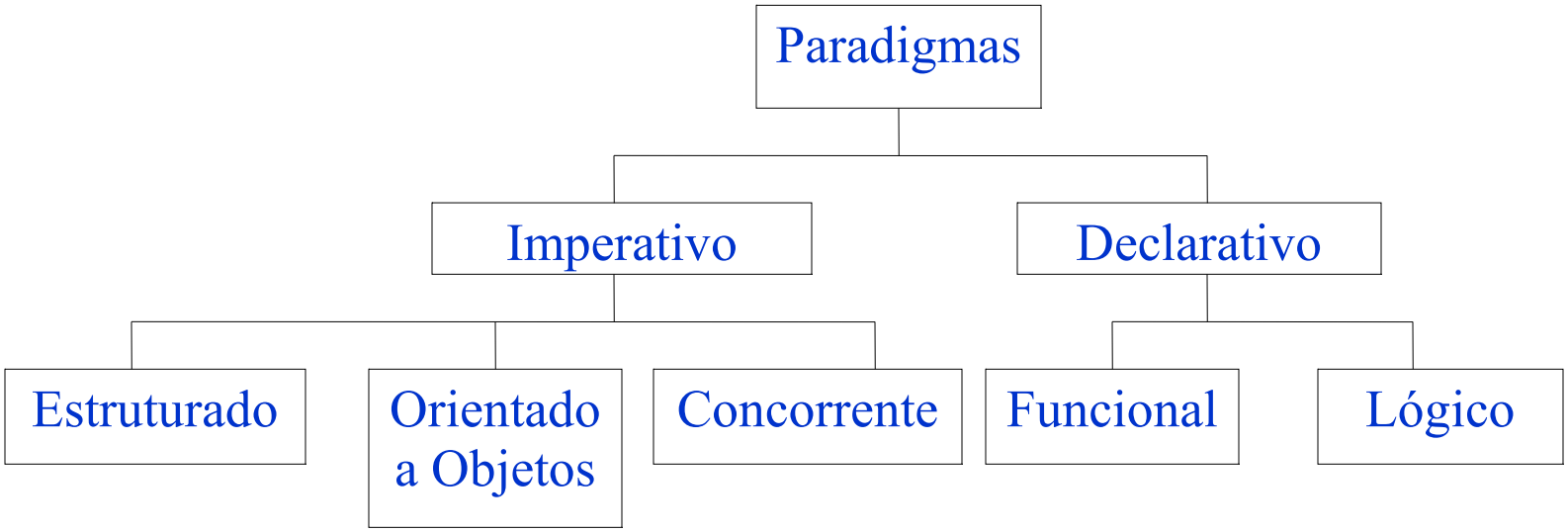
<sinal> ::= + | –

*<operador> ::= + | – | / | **

Gramática da linguagem

- **Semântica:** **Valor de “a” deve ser substituído pelo de “b” e retornado**

Paradigmas de LPs



Paradigmas de LPs

- Imperativo
  - Computação como 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 (GOTO)
  - Abstração de controle de execução

Pascal e C

- Orientado a Objetos
  - Abstração de Dados

Smaltalk, Java, C++, C#

- Concorrente
  - Processos executam Simultaneamente e Concorrem por recursos

ADA e Java

Paradigmas de LPs

- Declarativo
  - Especificações sobre a Tarefa a ser Realizada
  - Abstrai-se do “Como” o computador é implementado
- Funcional
  - Computação como Funções
- Lógico
  - Fatos e Regras sobre o domínio
  - Dedução Automática
  - Computação como Processo Lógico

Evolução de LPs

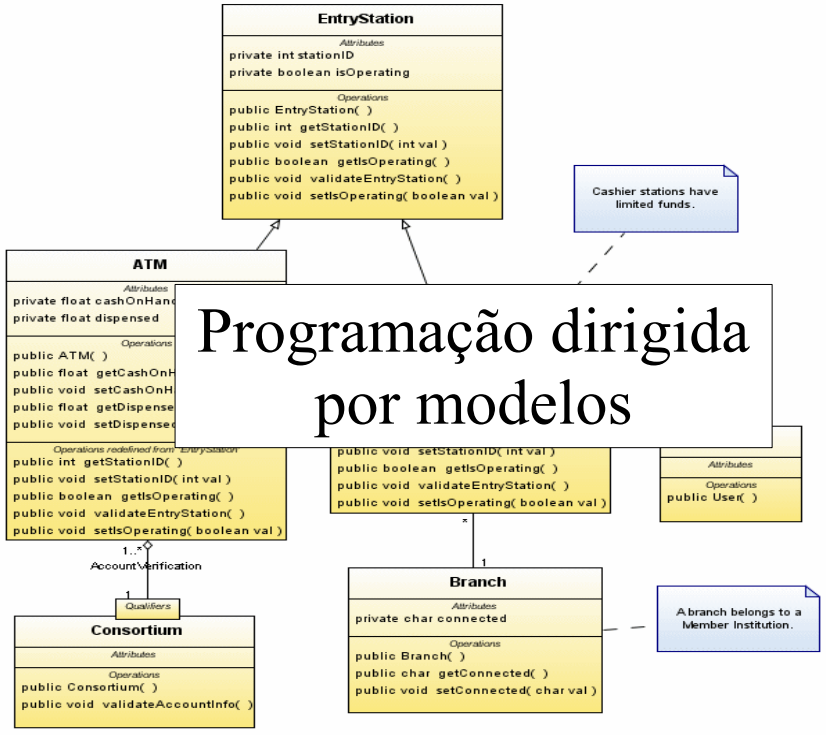
```
0400 2073FE JSR #FE73
0403 A200   LDX ##0
0405 BD800
0408 F006
040A 2075F
040D E8
040E D0F5
0410 00
0411 B9
0480 48
0481 45
0482 4C
0483 4C
0484 4F
0485 00
0486 67
```

```
program webtuga;
uses crt;

var contador,tabuada: integer;

begin
      end;
   readkey;

end.
```

CODIGO CEDIDO POR CENOURINHA - WWW.WEB

```
package lp2007;
import lp2006.*;
public class
```

Programação dirigida por modelos



Quero que você some esses dois valores: 3 e 4

ou

Calcule 3 + 4

ou

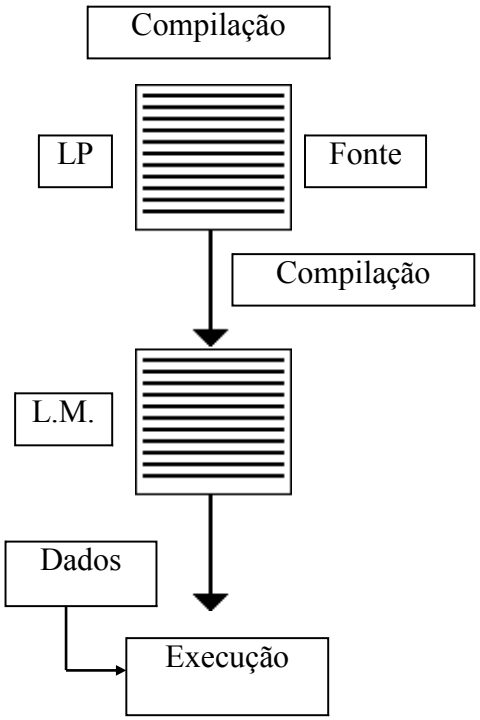
Adicione 3 a 4 e dê o resultado

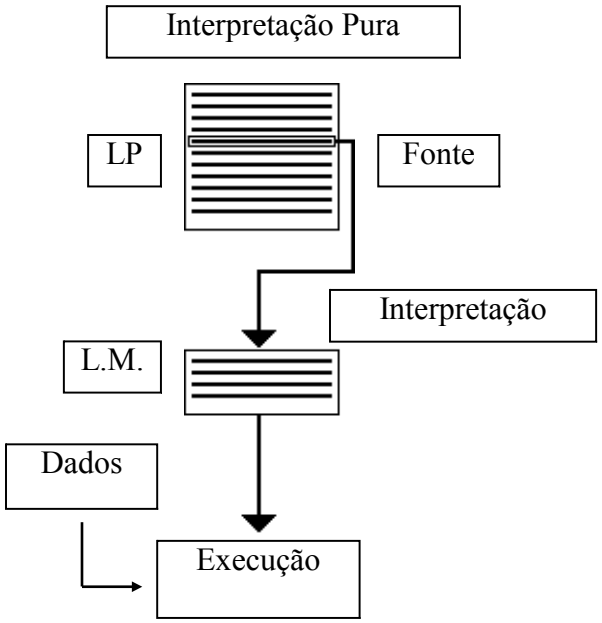
Trade-offs no projeto de linguagens

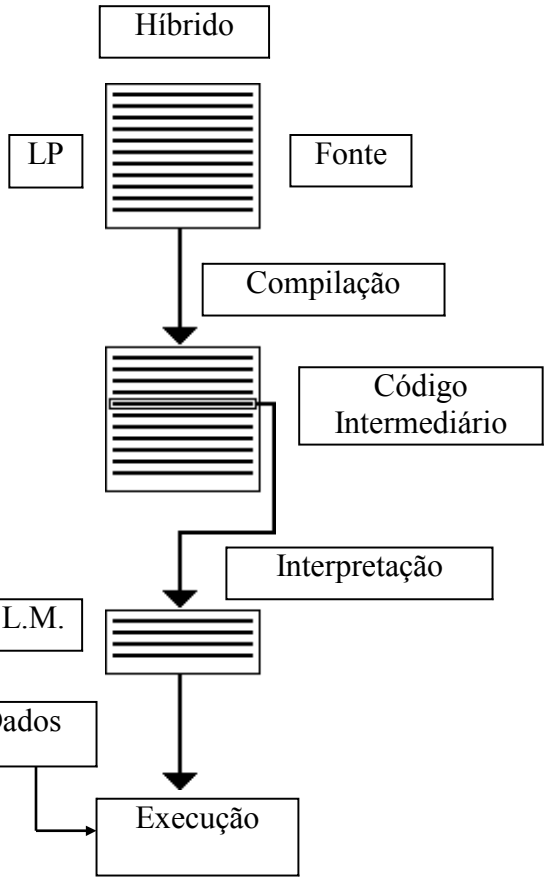
- Confiabilidade x Custo de execução
  - Verificação dinâmica dos índices de um array
- Legibilidade x Capacidade de escrita
  - APL possui vários operadores para manipular arrays, facilitando a expressão de fórmulas. A leitura, porém, é muito difícil
    - 4h para entender programa de 4 linhas
- Flexibilidade x Segurança
  - Aritmética de apontadores usando registros variantes (perigoso)

Implementação de LPs

- Todo programa precisa ser traduzido para a linguagem de máquina a fim de ser executado







Implementação de LP

- Compilação
  - Tradução do código fonte para a linguagem de máquina
  - Execução mais rápida
- Interpretação pura
  - Código fonte é interpretado sem nenhuma conversão
  - Execução lenta (de 10 a 100 vezes)
- Implementação híbrida
  - Código fonte é traduzido para uma linguagem intermediária a qual é interpretada

Just-in-Time (JIT)

- Traduzem parte do código intermediário (*byte code*) para linguagem de máquina
  - Trechos dentro de laços aninhados (executados inúmeras vezes)
  - **Trade-off:** Custo da tradução x redução no tempo de execução
- Suportado nas linguagens:
  - Java (apenas nas versões mais recentes)
  - Linguagens da plataforma .Net (C#, Visual Basic.Net, ...)

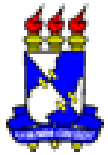
Ambiente de Programação

- Pode ser um critério de escolha da LP
- Coleção de ferramentas usadas no desenvolvimento
  - Sistema de arquivos
  - Editor de texto
  - Linker
  - Compilador
  - Ferramentas integradas
- Borland C++, Turbo Pascal, Delphi, JBuilder, VB, Visual Studio, NetBeans, Eclipse...

Histórico das Linguagens de Programação

Prof. Alberto Costa Neto
alberto@ufs.br

Linguagens de Programação



Departamento de Computação
Universidade Federal de Sergipe

Histórico de LPs

- FORTRAN (1957)
  - Aplicações numéricas e científicas (poucos dados e muita computação)
  - Ênfase em eficiência computacional
    - Única estrutura de controle era GOTO
    - Não havia alocação dinâmica de memória
- LISP (1959)
  - Programação funcional
  - Ênfase em processamento simbólico
  - Ainda hoje é a mais usada em IA

Histórico de LPs

- ALGOL (1960)
  - Programação estruturada
  - Primeira LP com sintaxe formalmente definida
  - Importância teórica (várias LP ALGOL-like)
- COBOL (1960)
  - Aplicações comerciais (muitos dados e pouca computação)
  - Focou legibilidade (similaridade com inglês) mas prejudicou a redigibilidade

Histórico de LPs

- BASIC (1964)
  - Ensino para leigos
- PASCAL (1971)
  - Ensino de programação estruturada
  - Simplicidade
- C (1972)
  - Ênfase na programação de sistemas (baixo nível)
  - Implementação de UNIX

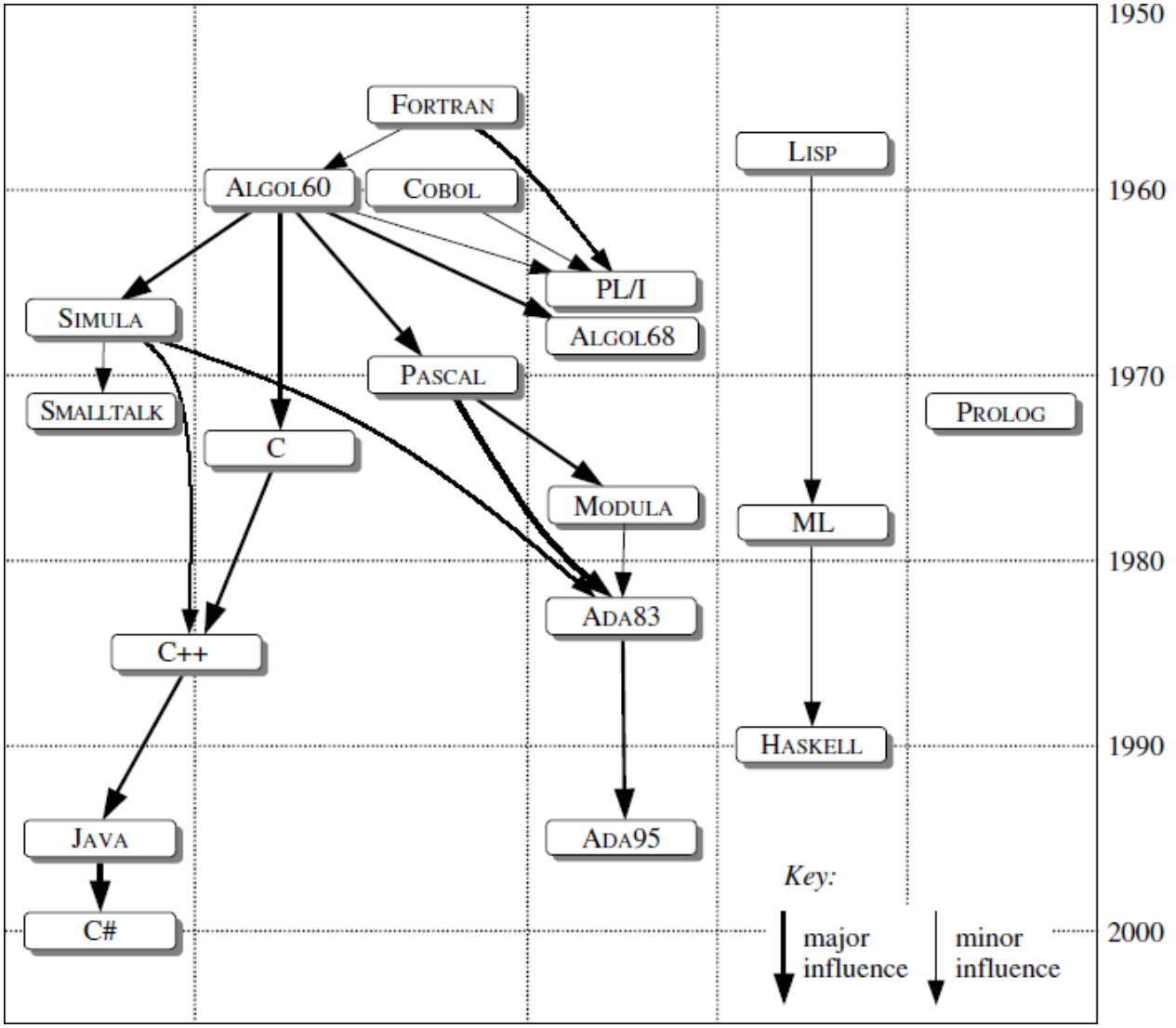
Histórico de LPs

- PROLOG (1972)
  - Programação lógica
  - Bastante uso em IA
- SMALLTALK (1972)
  - Programação orientada a objetos
  - Introduziu o conceito de GUI
- ADA (1983)
  - Grande e complexa (8 anos de desenvolvimento)
  - Programação concorrente e sistemas de tempo real

Histórico de LPs

- C++ (1985)
  - Foco em suporte à POO mantendo a eficiência de C
  - Disseminação da POO
- JAVA (1995)
  - Inicialmente criada para sistemas embarcados (set-up boxes, eletrodomésticos)
  - Baseada em C++ (facilidade de aprendizado)
  - Simples, confiável, portável
  - Internet

Histórico

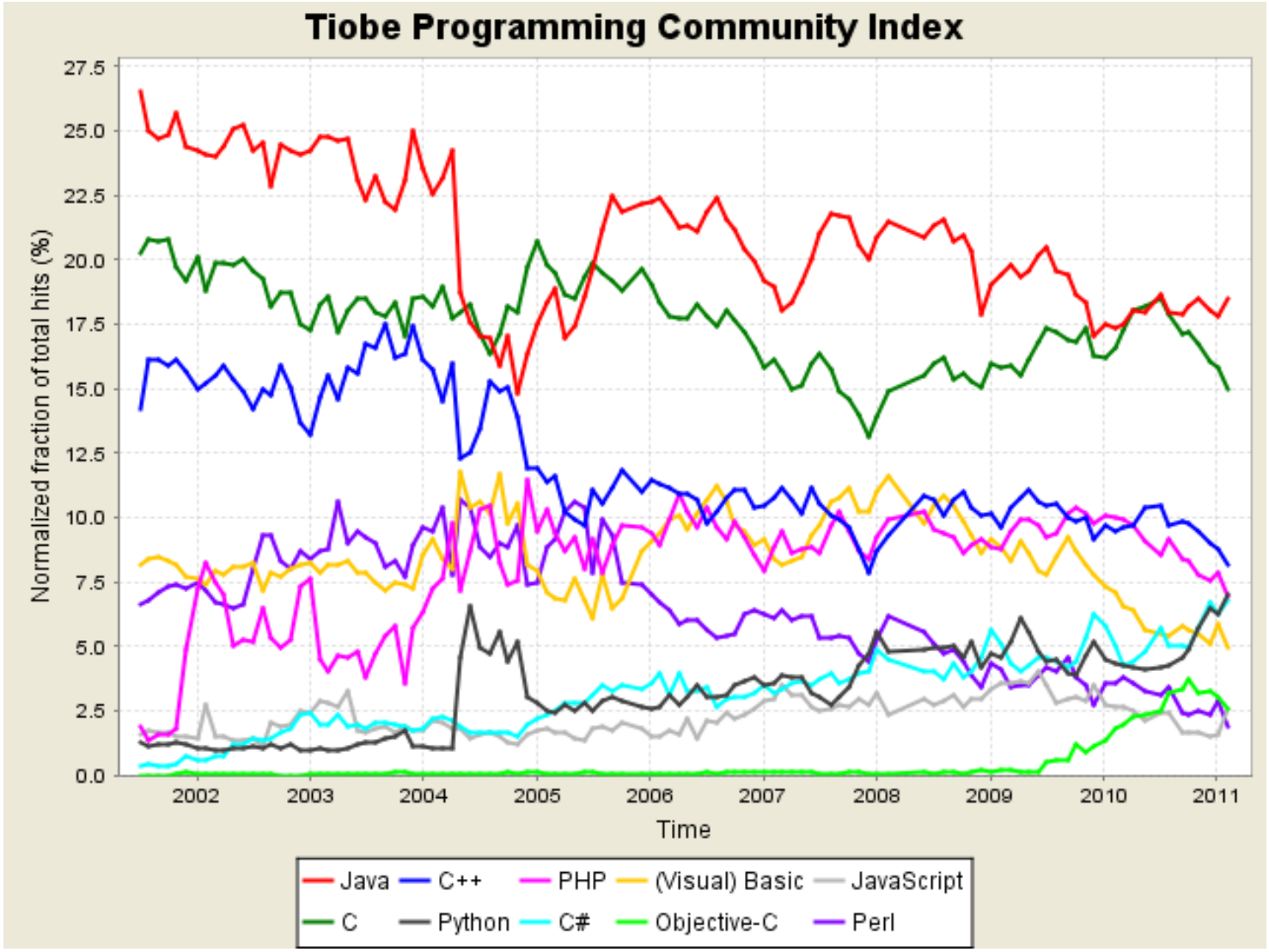

1950
1960
1970
1980
1990
2000
FORTRAN
LISP
ALGOL60
COBOL
PL/I
SIMULA
ALGOL68
PASCAL
PROLOG
SMALLTALK
C
MODULA
ML
ADA83
C++
HASKELL
JAVA
ADA95
C#
Key:
major influence
minor influence

TIOBE Index Fev - 2011

- Indicador de popularidade das LPs
- Atualizado mensalmente e baseado em:
  - Número de engenheiros, cursos e vendedores ao redor do mundo
  - Engenhos de busca populares são usados também para calcular as posições

Position Feb 2011	Position Feb 2010	Delta in Position	Programming Language	Ratings Feb 2011	Delta Feb 2010	Status
1	1	=	Java	18.482%	+1.13%	A
2	2	=	C	14.986%	-1.62%	A
3	4	↑	C++	8.187%	-1.26%	A
4	7	↑↑↑	Python	7.038%	+2.72%	A
5	3	↓↓	PHP	6.973%	-3.03%	A
6	6	=	C#	6.809%	+1.79%	A
7	5	↓↓	(Visual) Basic	4.924%	-2.13%	A
8	12	↑↑↑↑	Objective-C	2.571%	+0.79%	A
9	10	↑	JavaScript	2.558%	-0.08%	A
10	8	↓↓	Perl	1.907%	-1.69%	A
11	11	=	Ruby	1.615%	-0.82%	A
12	-	=	Assembly*	1.269%	-	A-
13	9	↓↓↓↓	Delphi	1.060%	-1.60%	A
14	19	↑↑↑↑↑	Lisp	0.956%	+0.39%	A
15	37	↑↑↑↑↑↑↑↑↑↑	NXT-G	0.849%	+0.58%	A--
16	30	↑↑↑↑↑↑↑↑↑↑	Ada	0.805%	+0.44%	A--
17	17	=	Pascal	0.735%	+0.13%	A
18	21	↑↑↑	Lua	0.714%	+0.21%	A--
19	13	↓↓↓↓↓	Go	0.707%	-1.07%	A--
20	32	↑↑↑↑↑↑↑↑↑↑	RPG (OS/400)	0.626%	+0.27%	A--

TIOBE (2001 - 2011)



Bibliografia (livros)

- Concepts of Programming Languages (Robert W. Sebesta)
  - Capítulos 1 e 2
- Programming Language Design Concepts (David Watt)
  - Capítulo 1
- Linguagens de Programação (Flávio Varejão)
  - Capítulo 1

Bibliografia (sites)

- TIOBE (http://www.tiobe.com)
- Histórico de LP (http://www.levenez.com/lang/)