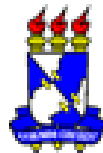


Paradigmas e Histórico de Linguagens de Programação

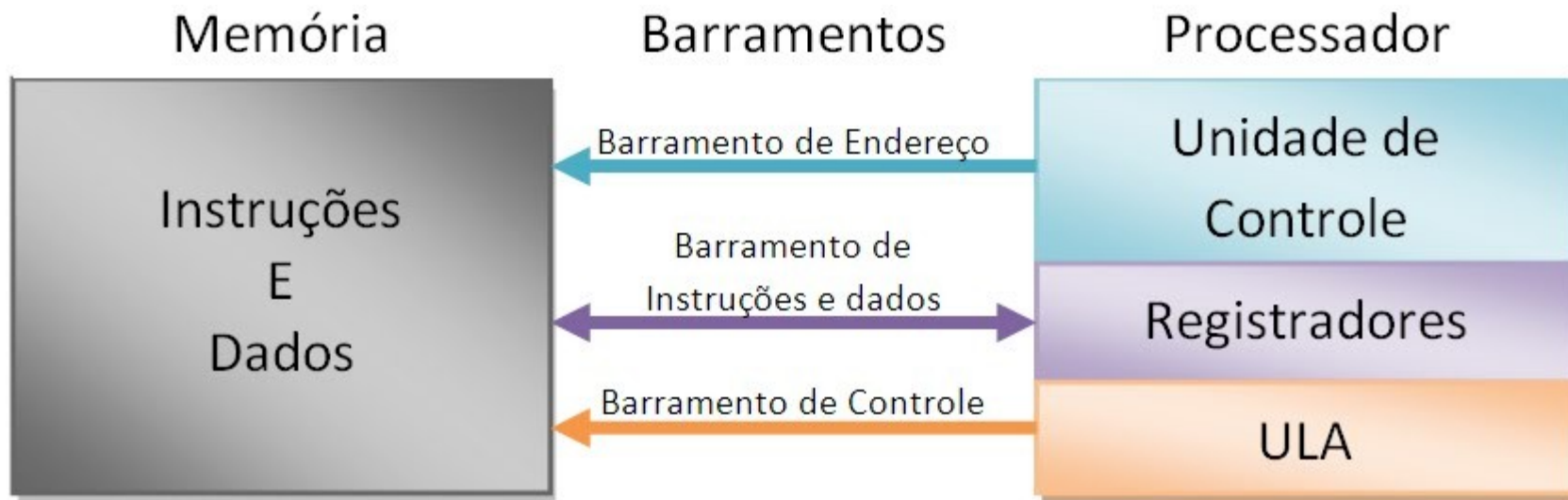
Prof. Alberto Costa Neto
alberto@ufs.br

Linguagens de Programação



Departamento de Computação
Universidade Federal de Sergipe

Influências sobre o projeto de linguagens



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



Evolução de LPs

```
11101000 10110010 00000110 00000000 00000000 01011001 01101000 10100000
10110000 01000000 00000000 01101010 00000000 11101000 11000010 10010111
00000000 00000000 10100 0400 2073FE JSR $FE73 s~ 1010
00000000 11101001 1000 0403 A200 LDX #$0 "□ 1111
00000111 00000000 0000 0405 BD800 program webtuga;
01000000 00000000 1100 0408 F006 uses crt;
11000011 11001100 1011 040A 2075F1 var contador,tabuada: integer;
11001001 01110100 0011 040D E8
00000000 00000000 0111 040E D0F5 begin
00000000 00000000 0000 0410 00
00000000 00000000 0000 0411 B9
01010000 11111111 0011 0480 48
11100100 10010111 0000 0481 45
00000000 00000000 0000 0482 4C
10000011 11111000 0000 0483 4C
00010101 11111111 0011 0484 4F
10100110 10010111 0000 0485 00
01010000 11101000 0110 0486 67
```

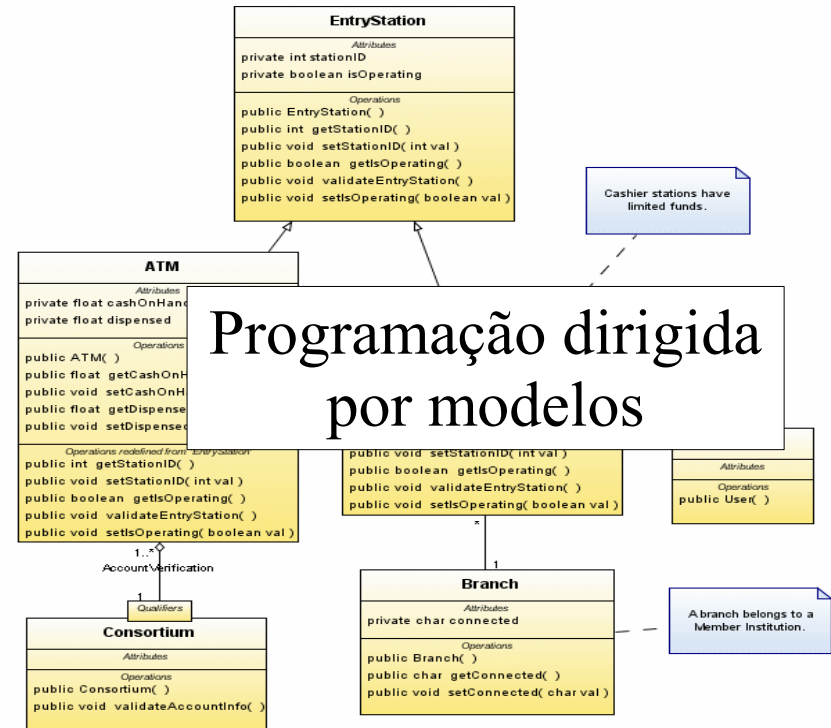
```
package lp2007;
import lp2006.*;
public class
```

```
end;
readkey;
```

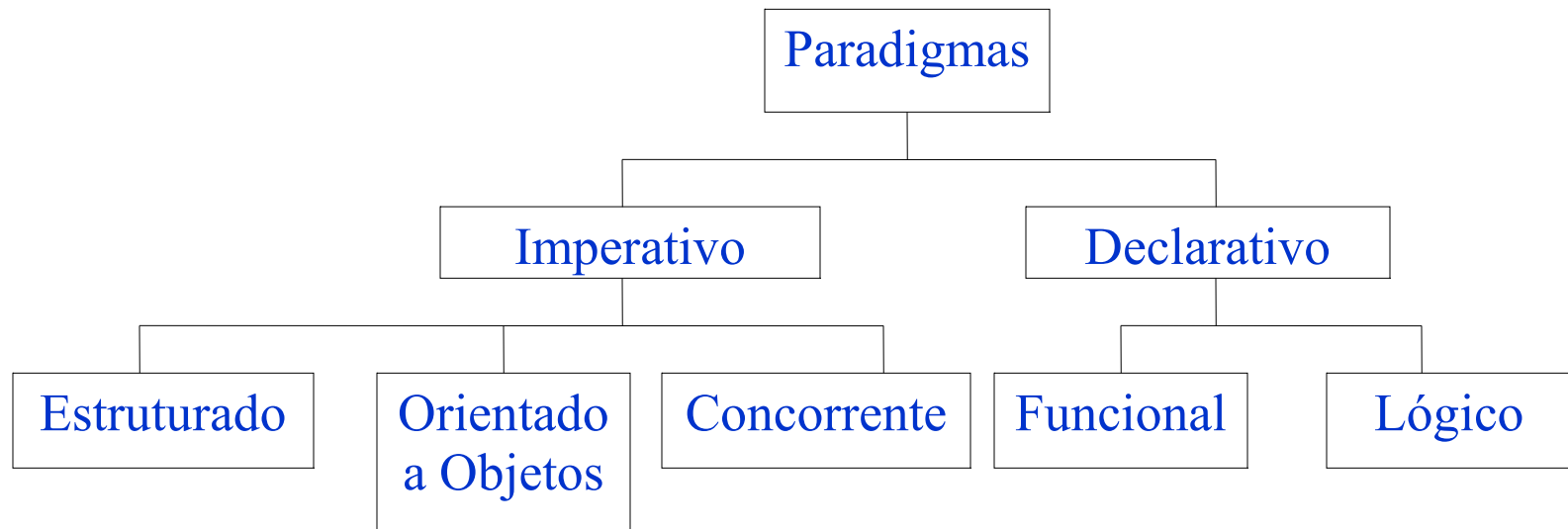
```
end.]
```

CODIGO Cedido POR CENOURINHA - WWW.WEB

Quero que você some esses
dois valores: 3 e 4
ou
Calcule $3 + 4$
ou
Adicione 3 a 4 e dê o resultado



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**
- Orientado a Objetos
 - **Abstração de Dados**
- Concorrente
 - Processos executam **Simultaneamente e Concorrem** por recursos

Pascal e C

Smaltalk, Java, C++, C#

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**



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)



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

$\langle \text{expressão} \rangle ::= \langle \text{valor} \rangle / \langle \text{valor} \rangle \langle \text{operador} \rangle \langle \text{expressão} \rangle$

$\langle \text{valor} \rangle ::= \langle \text{número} \rangle / \langle \text{sinal} \rangle \langle \text{número} \rangle$

$\langle \text{número} \rangle ::= \langle \text{semsinal} \rangle / \langle \text{semsinal} \rangle . \langle \text{semsinal} \rangle$

$\langle \text{semsinal} \rangle ::= \langle \text{dígito} \rangle / \langle \text{dígito} \rangle \langle \text{semsinal} \rangle$

$\langle \text{dígito} \rangle ::= 0 / 1 / 2 / 3 / 4 / 5 / 6 / 7 / 8 / 9$

$\langle \text{sinal} \rangle ::= + / -$

$\langle \text{operador} \rangle ::= + / - / / / *$

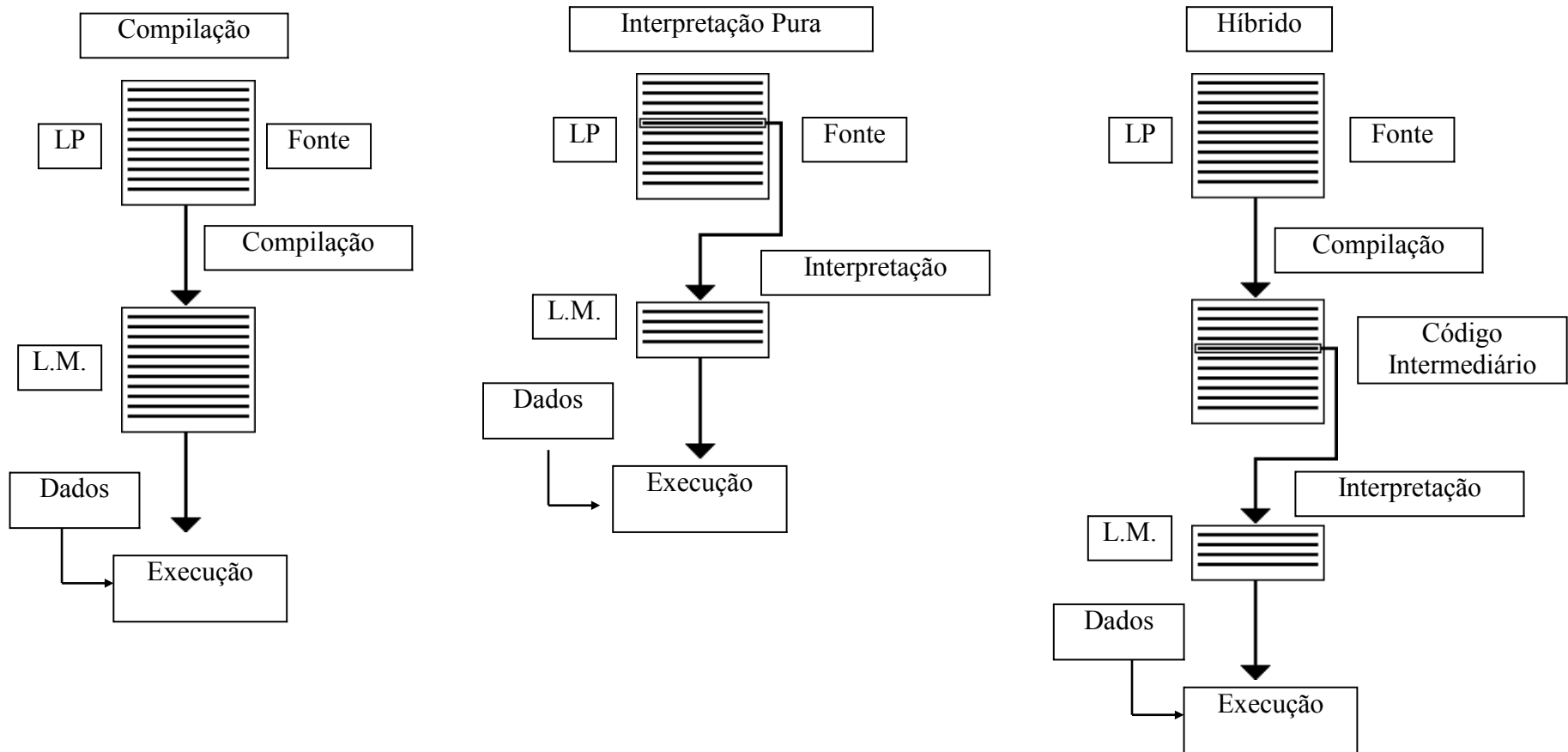
Gramática da linguagem

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



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 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

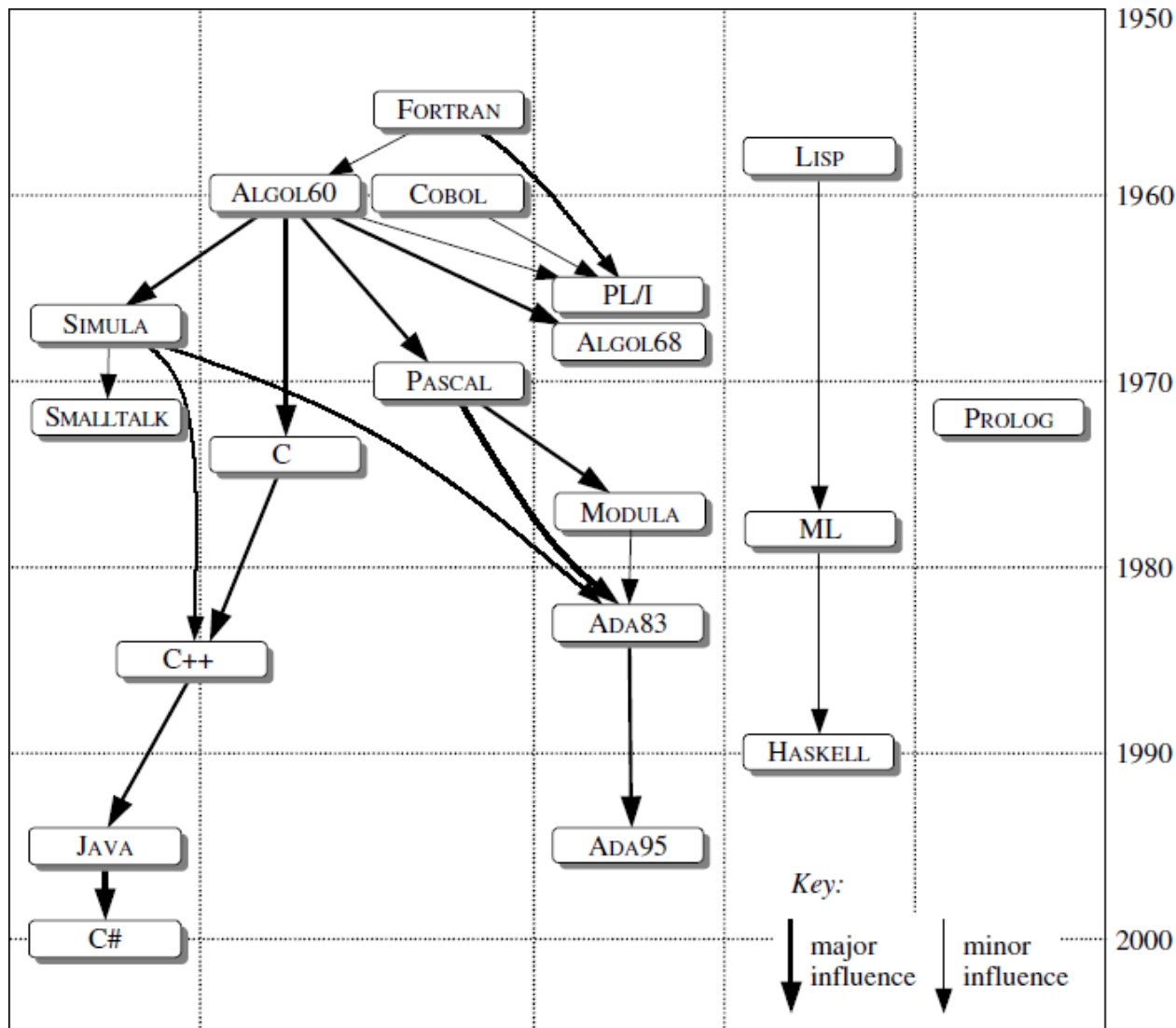


Histórico de LPs

- 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, portátil
 - Internet



Histórico



TIOBE Index

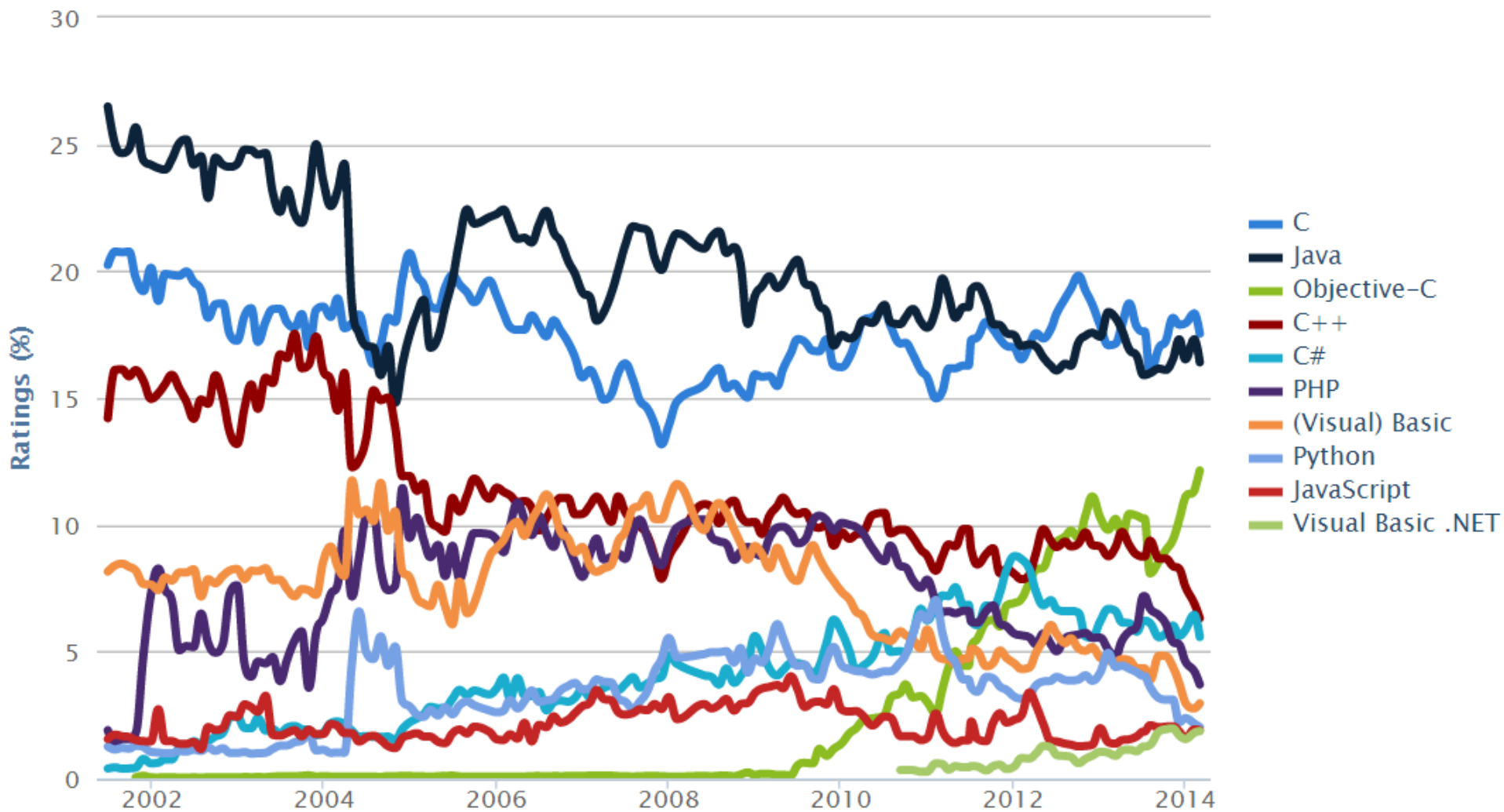
Março - 2014

- 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

Mar 2014	Mar 2013	Change	Prog Language	Ratings	Change
1	2	▲	C	17.535%	+0.39%
2	1	▼	Java	16.406%	-1.75%
3	3		Objective-C	12.143%	+1.91%
4	4		C++	6.313%	-2.80%
5	5		C#	5.572%	-1.02%
6	6		PHP	3.698%	-1.11%
7	7		(Visual) Basic	2.955%	-1.65%
8	8		Python	2.021%	-2.37%
9	11	▲	JavaScript	1.899%	+0.53%
10	16	▲▲	Visual Basic .NET	1.862%	+0.97%
11	17	▲▲	Transact-SQL	1.477%	+0.64%
12	69	▲▲	F#	1.216%	+1.14%
13	10	▼	Perl	1.149%	-0.81%
14	9	▼▼	Ruby	0.974%	-1.18%
15	15		Delphi/Object Pascal	0.881%	-0.01%
16	13	▼	Lisp	0.747%	-0.19%
17	30	▲▲	D	0.744%	+0.40%
18	20	▲	Assembly	0.732%	+0.10%
19	14	▼▼	PL/SQL	0.713%	-0.21%
20	23	▲	MATLAB	0.688%	+0.07%



TIOBE (2001 - 2014)



Bibliografia (livros)

- Concepts of Programming Languages
(Robert W. Sebesta)
 - Capítulo 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/>)

