

Introdução

**Linguagens de Programação
2019.2020**

Teresa Gonçalves
tcg@uevora.pt

Departamento de Informática, ECT-UÉ

Sumário

O que é uma LP?

Porquê estudar LP?

Desenho de uma LP

Paradigmas de LP

CrITÉrios de avaliação de uma LP

Métodos de implementação

Curiosidades

O que é uma LP?

8 “cientistas da computação”, 8 respostas

“notação formal para computações”

“ferramenta para escrever programas”

“forma de comunicação entre programadores”

“veículo de expressão para desenhos de alto nível”

“notação para algoritmos”

“forma de expressar relações entre conceitos”

“ferramenta para experimentar soluções para problemas”

“forma para controlar dispositivos computadorizados”

Conhecimentos de um Informático

Conceitos para resolução de problemas e algoritmos

Técnicas para desenvolver aplicações grandes

Funcionamento da máquina e sistema operativo

Detalhes sintáticos relevantes de uma linguagem

Características fundamentais dos paradigmas de linguagens

Melhor paradigma para cada tipo de problema / tarefa

Porquê estudar LP?

Aumentar aptidão para expressar ideias

Novos paradigmas e formas de pensar

Melhorar conhecimentos para escolher linguagens apropriadas

Aumentar aptidão para aprender novas linguagens

Perceber a importância da implementação

Porque é que a recursividade é lenta?

Como são implementadas as estruturas de dados?

Conhecer o avanço geral da computação

Desenho de uma linguagem

Abstrações do computador (1)

Linguagem máquina

Fornece instruções primitivas

Sistema operativo

Fornece primitivas de abstração superior

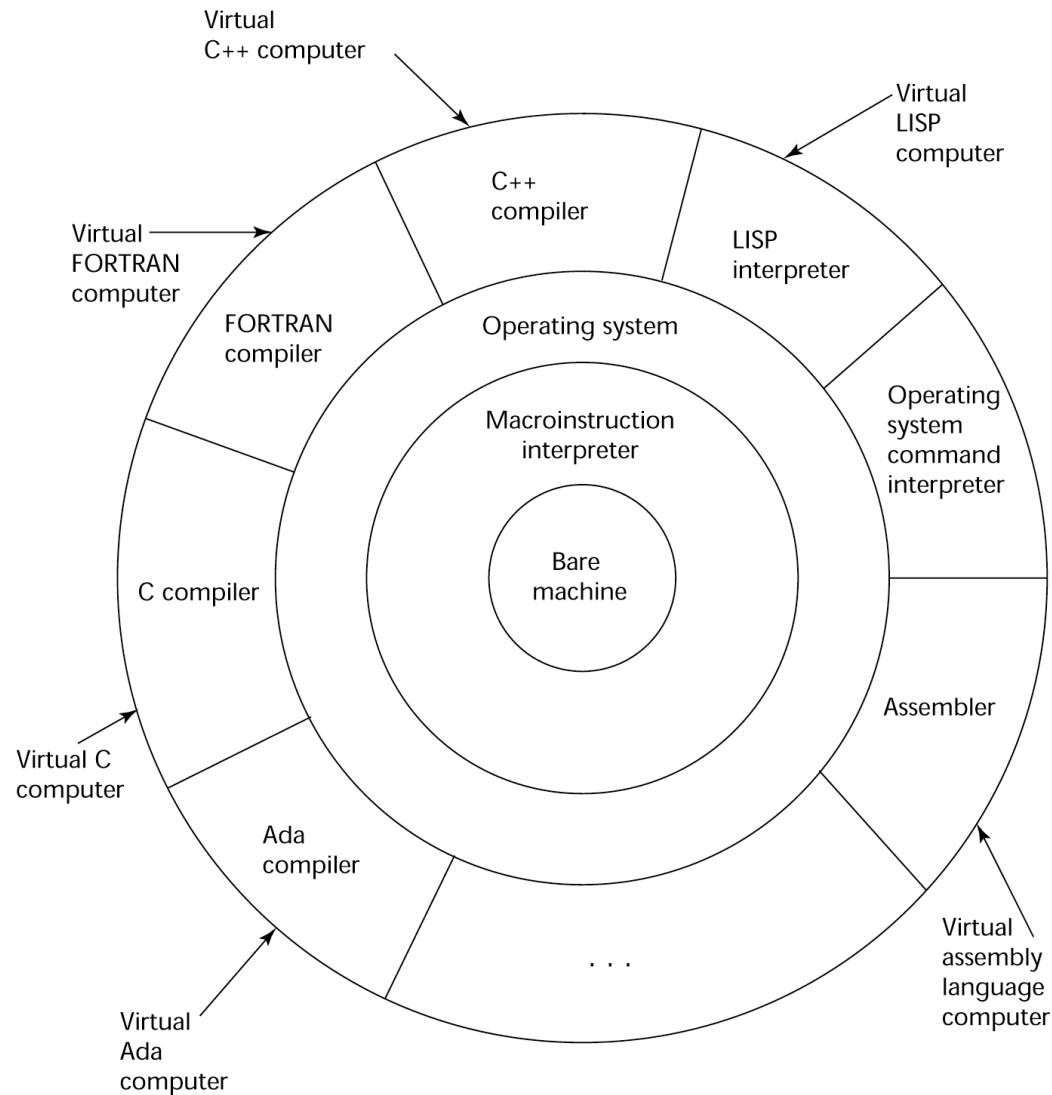
Gestão de recursos, input/output, gestão de ficheiros, editores de texto, etc.

Implementação de linguagem

Fornece um **computador virtual**

Serve de interface entre o computador e a máquina virtual (sistema operativo+máquina)

Abstrações do computador (2)



Influências no desenho de uma LP

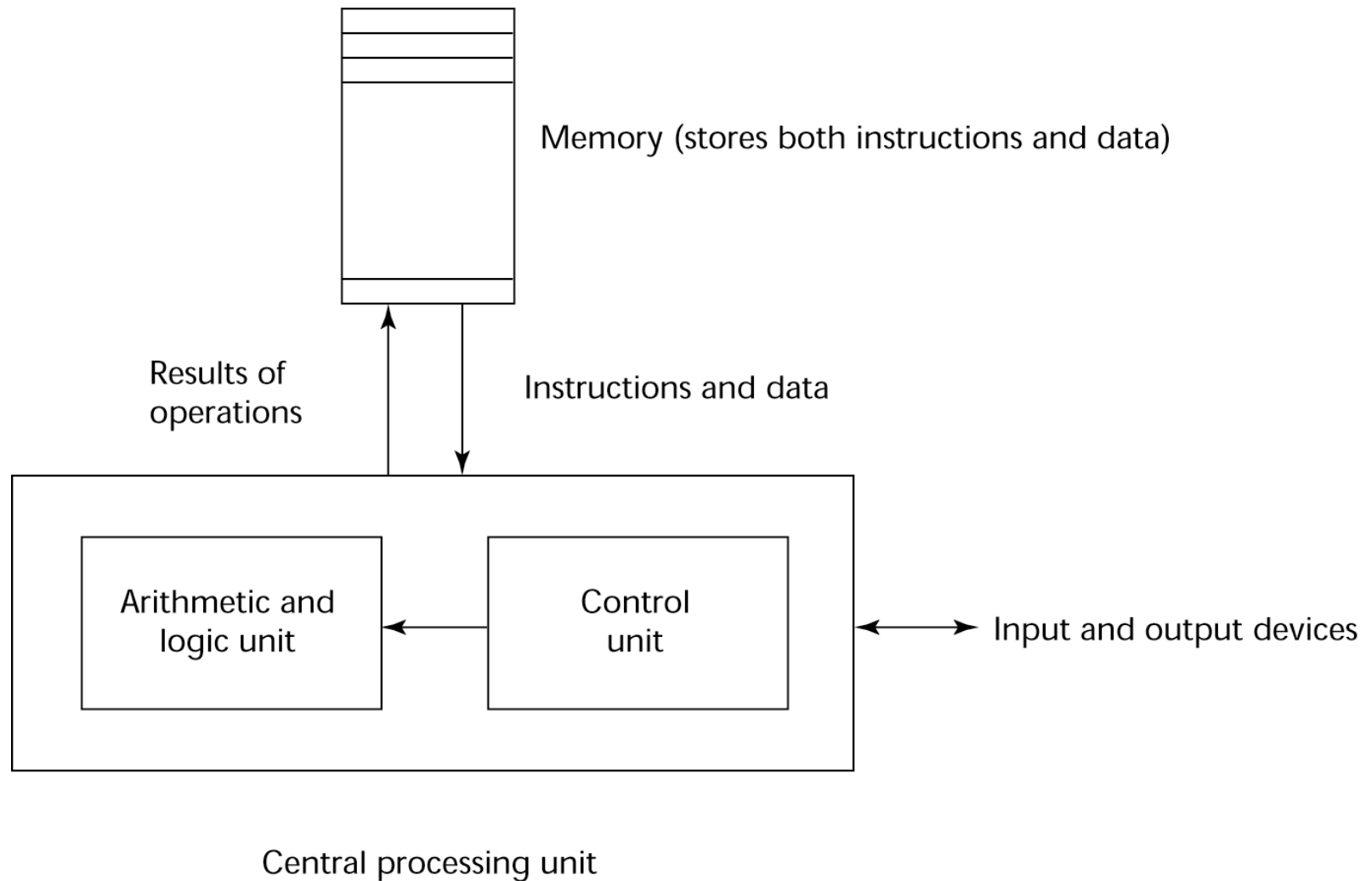
Arquitetura do computador

Linguagens imperativas

Domínio de aplicação da LP

Metodologias de programação

Arquitetura de von Neumann



Implicações

Desenvolvimento de linguagens imperativas

Variável

modela posições de memória

Atribuição

modela operações de *piping* (transferência entre CPU e memória)

Iteração é eficiente

As instruções são guardadas em posições adjacentes de memória e os ciclos requerem uma instrução de salto simples

Desencoraja a utilização da recursividade

Domínios de aplicação

Aplicações científicas

Grande número de cálculos em vírgula flutuante (Fortran)

Aplicações de negócios

Produção de relatórios, utilização de números decimais e caracteres (COBOL)

Inteligência Artificial

Manipulação de símbolos em vez de números (Lisp)

Programação de Sistemas

Eficiência necessária devido à utilização contínua (C, C++)

Linguagens de *Scripting*

Lista de comandos que devem ser executados (Perl, JavaScript, PHP)

Lings. com finalidades específicas

(SNOBOL, APL)

Metodologias de programação (1)

Anos 50

Aplicações simples

Eficiência da máquina

assembly

Finais anos 60

Aplicações mais complexas

Orientação a processos (eficiência das pessoas: legibilidade, melhores estruturas de controlo)

Programação estruturada

Desenho top-down, com refinamento

Verificação de tipos incompleta e controlo “pobre”

Metodologias de programação (2)

Finais anos 70

Orientação a dados

Abstração de dados (EDA, módulos)

SIMULA 67: 1ª ling com suporte de dados abstratos

Anos 80s

Orientação a objetos

Encapsulamento, herança, ligação dinâmica de métodos

SMALLTALK: 1ª ling OO pura (1989)

Suporte fácil em linguagens imperativas (C++, ADA95, JAVA)

LISP (CLOS 1988) e Prolog (Prolog++ 1994)

Metodologias de programação (3)

Atualidade

Concorrência

Arquitetura multi-processador

Suporte em JAVA, ADA, Python

Paradigmas de uma LP

Paradigmas

Procedimental / Programação Imperativa

Funcional / Programação Aplicativa

Declarativa / Programação Lógica

Programação Orientada a Objetos

Procedimental

Instruções são comandos

“Adiciona 17 a x”

Características

Muito próxima da arquitetura von Neumann

Operações chave

Atribuição

Iteração

Linguagens

C, Pascal, ADA

Funcional

Instruções descrevem o valor de expressões

“O reverso de uma lista é o último elemento seguido do reverso do resto da lista”

Operações chave

Avaliação de expressões através da aplicação de funções
Recursividade

Linguagens

LISP, Scheme, ML, Haskell

Declarativa

Instruções descrevem factos e regras

Facto: “O João é o pai do Nuno.”

Regra: “Se x é pai de y e y é pai de z , x é avô de z .”

Características

O programa não diz como encontrar a solução

Operações chave

Unificação

Linguagens

Prolog

Orientada a Objetos

Descreve a comunicação entre objectos

“Fracção f1, simplifica-te.”

Características

Pode ser imperativa ou funcional

“Empacota” dados com processamento

Operações chave

Passagem de mensagens

Herança e ligação dinâmica de tipos

Linguagens

Smalltalk, SIMULA, C++, Java, CLOS

Outros

Paradigmas híbridos

- Concorrente

- Paralelo

- Fluxo de dados

- Específico (para domínios particulares)

Linguagens de marcação

- Características

 - Não especificam cálculos → não são LP

 - Desenho, avaliação e implementação similar a LP

- Linguagens

 - HTML, XML

Critérios de avaliação de uma LP

Critérios de avaliação (1)

Legibilidade

Simple e ortogonal

Instruções de controlo

Definição de tipos de dados / estruturas

Afecta a facilidade de manutenção!

Facilidade de escrita

Simple e ortogonal

Suporte para abstracção

Expressiva

Dependente do domínio!

Critérios de avaliação (2)

Fiabilidade

Verificação de tipos

Tratamento de exceções

Legibilidade e facilidade de escrita

Custo

Tipos de custo

Desenvolvimento do programa

Manutenção

Fiabilidade

Depende da legibilidade e facilidade de escrita!

Compromissos

Legibilidade vs. Facilidade de escrita

Variedade de funções (APL, PERL)

A inclusão aumenta facilidade de escrita, mas diminui a legibilidade

Fiabilidade vs. custo de execução

Verificação dos limites dos arrays (Java)

inclusão aumenta a fiabilidade, mas diminui a eficiência

Flexibilidade vs. Segurança

Registos variantes

Flexível, mas perigoso em tempo de execução

Métodos de implementação de uma LP

Métodos de implementação (1)

Compilação

Com funciona?

Traduz o programa de alto nível para código-máquina

Características

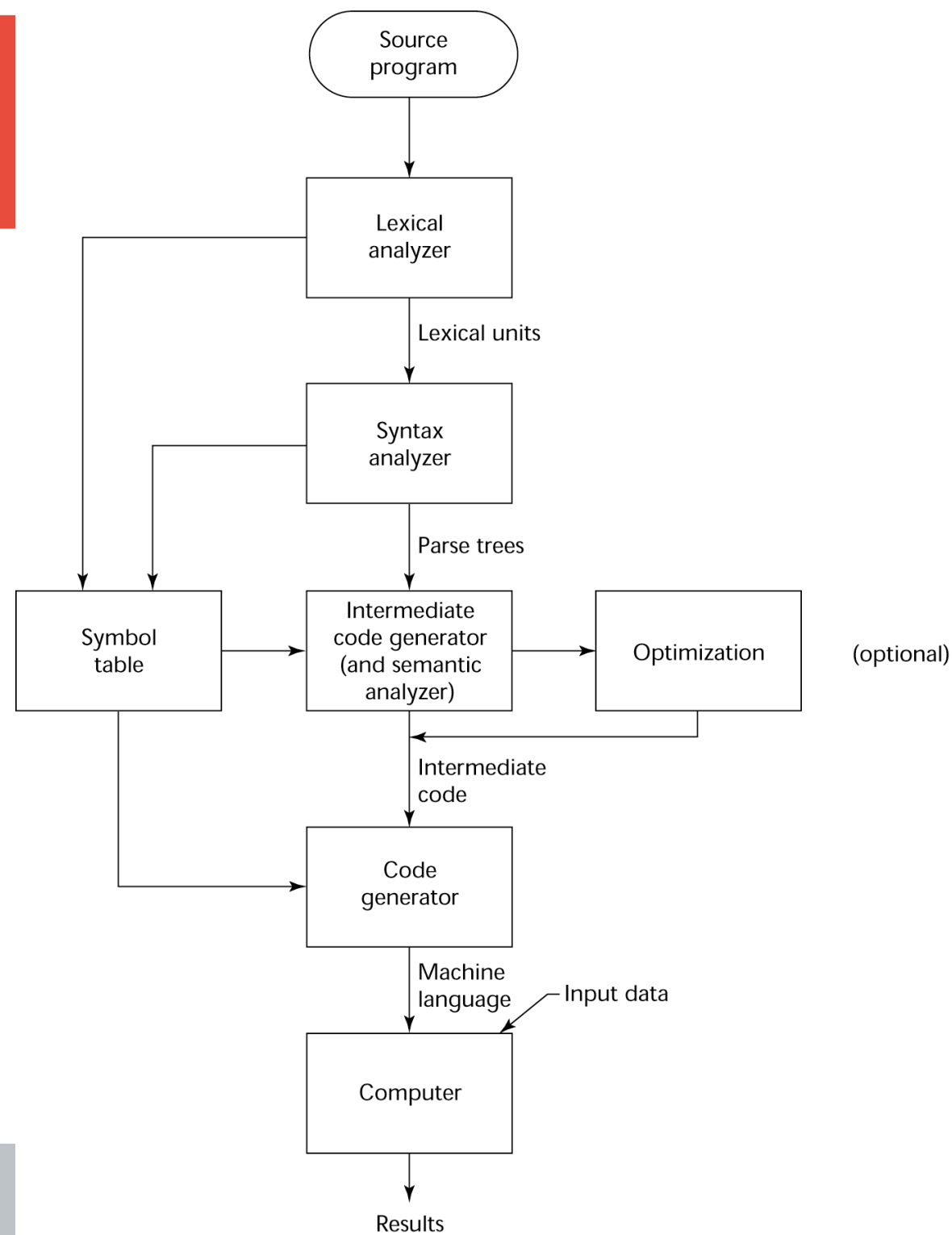
Tradução lenta

Execução rápida

Linguagens

maioria das LP de produção (C, C++, Ada, COBOL, Pascal)

Processo de compilação



Métodos de implementação (2)

Interpretação pura

Como funciona?

Não há tradução

Programas são interpretados por outro programa

Características

Execução lenta (10-100x mais lenta)

Sempre que uma linha é executada é decodificada primeiro

Requer mais espaço durante a execução

A tabela de símbolos e o código fonte devem estar disponíveis

Debug mais fácil

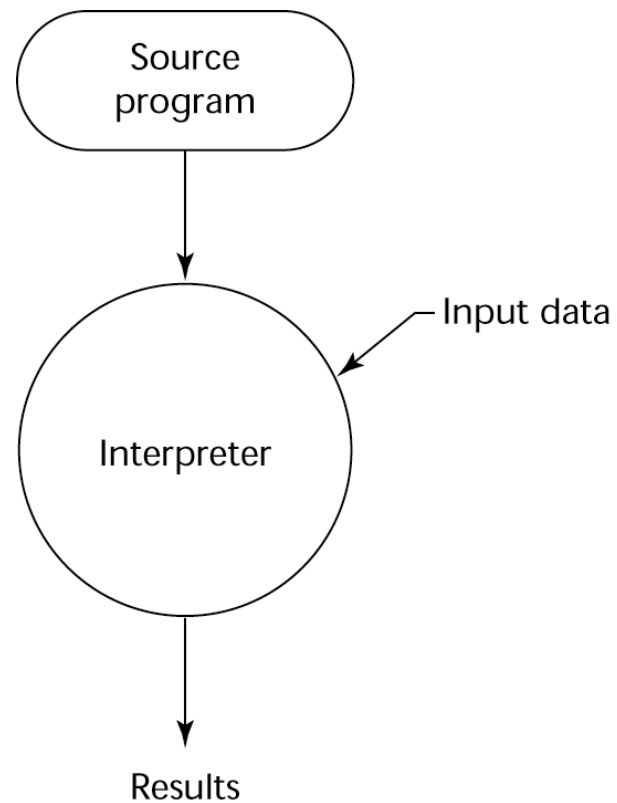
Erros ocorrem logo que a linha é executada

Linguagens

Primeiras versões de LISP, APL, SNOBOL

JavaScript, PHP

Interpretação Pura



Métodos de implementação (3)

Sistema híbrido

Como funciona?

Traduz os programas de linguagem de alto-nível para uma linguagem intermédia

Esta linguagem é depois interpretada

Características

Custo de tradução baixo

Tempo de execução médio

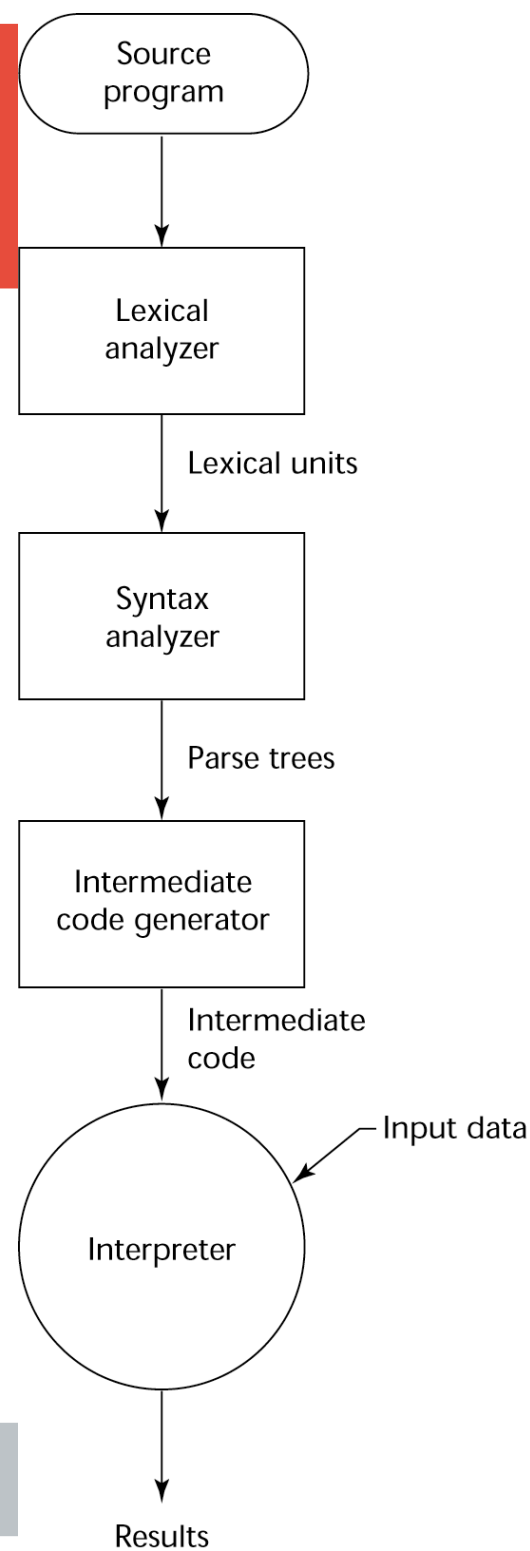
Mais rápido que a interpretação pura porque o código fonte é descodificado uma única vez

Linguagens

PERL, JAVA (utilização da VM)

Algumas aplicações JAVA já são compiladas

Sistema híbrido



Ambientes de programação

Coleção de ferramentas para desenvolvimento de software

UNIX

Ferramentas separadas: editor, compilador, debugger, linker

Microsoft Visual Studio.NET

Ambiente visual

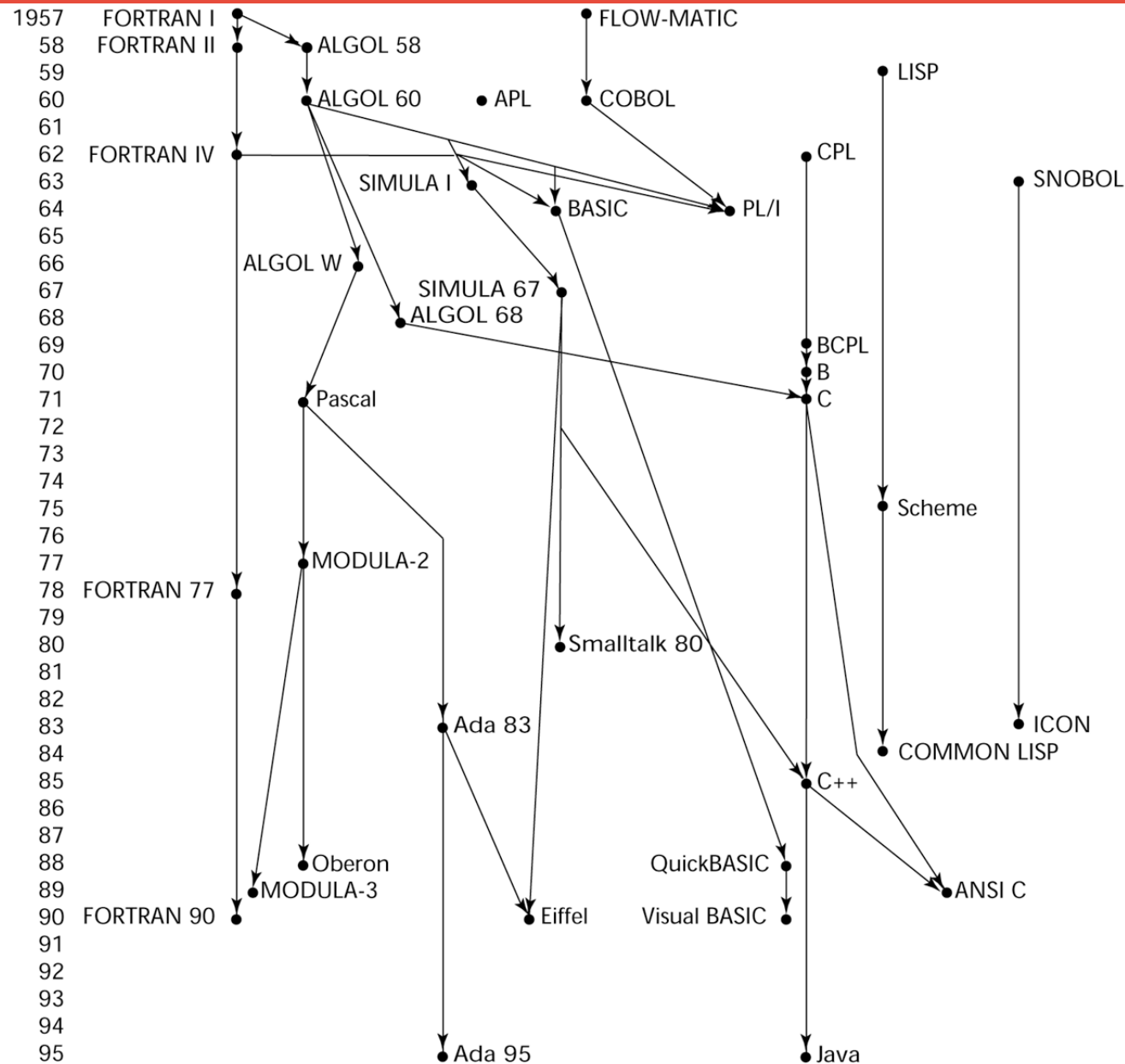
Utilizado para programar em C#, Visualbasic.NET, Jscript, J# ou C++

ECLIPSE

Ambiente open-source

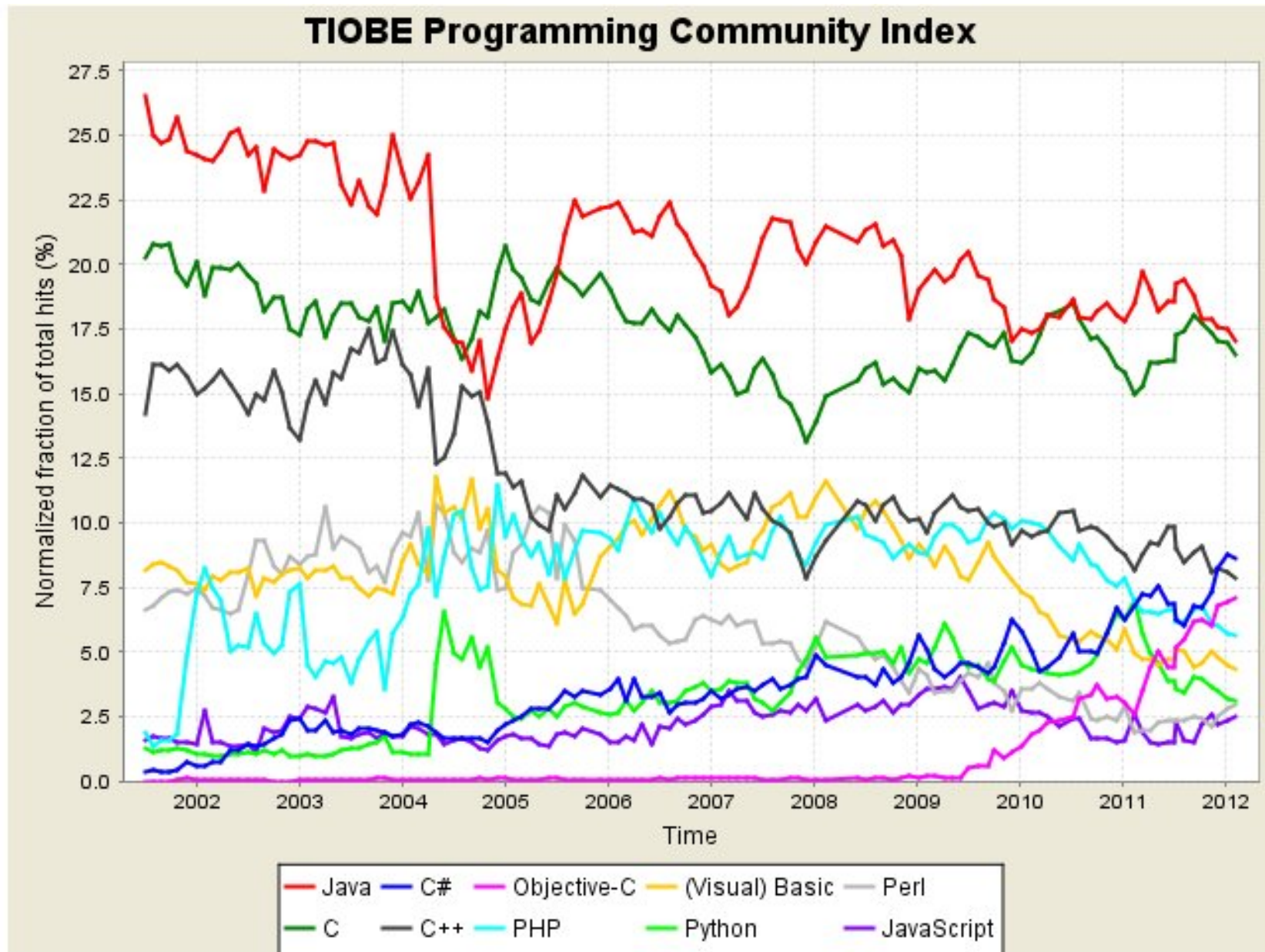
Curiosidades

Genealogia das LP



pretty-one

Evolução



Popularidade (2013)

Language Popularity Index - Web queries done on: 2013/02/01 22:15

<http://lang-index.sourceforge.net/#grid>

Language category: any *) 123 entries.					Language category: general-purpose *) 48 entries.			Language category: script *) 49 entries.			Language category: other *) 26 entries.		
Rank	Name	Share	Last month's share	Last year's share	Rank	Name	Share	Rank	Name	Share	Rank	Name	Share
1	C	17.836%	17.780%	18.257%	1	C	24.725%	1	PHP	19.093%	1	COBOL	14.162%
2	Java	17.057%	15.031%	17.707%	2	Java	23.644%	2	Python	16.321%	2	Logo	13.540%
3	Objective-C	10.159%	9.962%	6.509%	3	Objective-C	14.083%	3	Perl	10.208%	3	SAS	11.594%
4	Basic	6.366%	6.218%	9.060%	4	Basic	8.824%	4	Ruby	6.880%	4	PL/SQL	11.068%
5	C++	6.356%	6.116%	6.032%	5	C++	8.811%	5	JavaScript	5.820%	5	Prolog	9.779%
6	C#	4.592%	6.544%	3.638%	6	C#	6.365%	6	R	5.459%	6	ABAP	5.685%
7	PHP	4.415%	4.604%	6.316%	7	Pascal	1.608%	7	NXT-G	4.743%	7	LabView	4.808%
8	Python	3.774%	4.409%	3.881%	8	Ada	1.233%	8	Bourne shell	4.030%	8	RPG (OS/400)	4.677%
9	Perl	2.360%	2.800%	3.820%	9	D	1.162%	9	MATLAB	2.783%	9	Focus	4.623%
10	Ruby	1.591%	1.580%	1.619%	10	Go	0.972%	10	Lisp/Scheme	2.555%	10	VHDL	3.245%
11	JavaScript	1.346%	1.269%	2.437%	11	Fortran	0.941%	11	Lua	2.314%	11	MUMPS	3.169%
12	R	1.262%	1.183%	1.380%	12	Delphi	0.934%	12	Scratch	1.918%	12	SIGNAL	1.924%
13	Pascal	1.160%	1.138%	1.206%	13	Haskell	0.694%	13	APL	1.702%	13	Transact-SQL	1.840%
14	NXT-G	1.097%	1.105%	0.134%	14	Smalltalk	0.570%	14	ABC	1.368%	14	Cg (Nvidia)	1.591%
15	Bourne shell	0.932%	0.880%	0.077%	15	Caml/F#	0.552%	15	Awk	1.253%	15	Progress	1.358%
16	Ada	0.889%	0.904%	0.891%	16	Scala	0.496%	16	J	1.097%	16	Verilog	1.316%
17	D	0.838%	0.807%	1.291%	17	Forth	0.476%	17	VBScript	0.891%	17	Natural	1.300%
18	Go	0.701%	0.654%	0.568%	18	ML	0.455%	18	Alice	0.734%	18	XSLT	1.250%
19	Fortran	0.679%	0.681%	0.604%	19	Erlang	0.446%	19	ActionScript	0.716%	19	Avenue	0.565%
20	Delphi	0.674%	0.685%	1.456%	20	Eiffel	0.323%	20	Clojure	0.691%	20	LabWindows/CVI	0.501%
21	COBOL	0.671%	0.643%	0.531%	21	PL/I	0.288%	21	Groovy	0.689%	21	XQuery	0.405%
22	MATLAB	0.644%	0.627%	0.501%	22	Icon	0.241%	22	IDL	0.612%	22	YACC	0.402%

Popularidade

Worldwide, Feb 2018 compared to a year ago:

Rank	Change	Language	Share	Trend
1		Java	22.55 %	-1.1 %
2		Python	21.3 %	+5.6 %
3		PHP	8.53 %	-1.8 %
4	↑	Javascript	8.49 %	+0.4 %
5	↓	C#	8.06 %	-0.6 %
6		C	6.51 %	-1.4 %
7	↑	R	4.23 %	+0.5 %
8	↓	Objective-C	3.86 %	-1.2 %
9		Swift	3.09 %	-0.4 %
10		Matlab	2.34 %	-0.5 %
11		Ruby	1.8 %	-0.4 %
12	↑↑↑	TypeScript	1.47 %	+0.5 %
13		VBA	1.46 %	+0.0 %
14	↓↓	Visual Basic	1.3 %	-0.3 %
15	↓	Scala	1.24 %	+0.1 %
16	↑↑↑↑↑↑	Kotlin	0.84 %	+0.7 %
17	↓	Perl	0.81 %	-0.1 %
18		Go	0.75 %	+0.2 %

Worldwide, Feb 2019 compared to a year ago:

Rank	Change	Language	Share	Trend
1	↑	Python	26.42 %	+5.2 %
2	↓	Java	21.2 %	-1.3 %
3	↑	Javascript	8.21 %	-0.3 %
4	↑	C#	7.57 %	-0.5 %
5	↓↓	PHP	7.34 %	-1.2 %
6		C/C++	6.23 %	-0.3 %
7		R	4.13 %	-0.1 %
8		Objective-C	3.04 %	-0.8 %
9		Swift	2.56 %	-0.6 %
10		Matlab	1.98 %	-0.4 %
11	↑↑	TypeScript	1.61 %	+0.2 %
12	↓	Ruby	1.54 %	-0.2 %
13	↓	VBA	1.44 %	-0.0 %
14	↑	Scala	1.17 %	-0.1 %
15	↑	Kotlin	1.15 %	+0.3 %
16	↓↓	Visual Basic	1.15 %	-0.1 %
17	↑	Go	1.05 %	+0.3 %
18	↓	Perl	0.58 %	-0.2 %

Tendências

<http://pypl.github.io/PYPL.html>

Worldwide, Python is the most popular language, Python grew the most in the last 5 years (16.2%) and PHP lost the most (-5.6%)

PYPL Popularity of Programming Language

