Critérios de Avaliação de LPs

ERICK GALANI MAZIERO
erick.maziero@ufla.br

Departamento de Ciências da Computação
Universidade Federal da Lavras

1 LPs mais usadas

Índice TIOBE

- http://tiobe.com/tiobe-index/
- É um indicador de popularidade de linguagens de programação
  - Atualizado mensalmente
  - Baseado em sistemas de busca mundiais
    - Google, Bing, Yahoo!, Wikipedia, Amazon, Youtube e Baidu

Índice TIOBE

Oct 2020	Oct 2019	Change	Programming Language	Ratings	Change
1	2	↑	C	16.95%	+0.77%
2	1	↓	Java	12.56%	-4.32%
3	3		Python	11.28%	+2.19%
4	4		C++	6.94%	+0.71%
5	5		C#	4.16%	+0.30%
6	6		Visual Basic	3.97%	+0.23%
7	7		JavaScript	2.14%	+0.06%
8	9	↑	PHP	2.09%	+0.18%
9	15	⇈	R	1.99%	+0.73%
10	8	↓	SQL	1.57%	-0.37%
11	19	⇈	Perl	1.43%	+0.40%
12	11	↓	Groovy	1.23%	-0.16%
13	13		Ruby	1.16%	-0.16%
14	17	↑	Go	1.16%	+0.06%

Índice TIOBE

Very Long Term History

To see the bigger picture, please find below the positions of the top 10 programming languages of many years back. Please note that these are *average* positions for a period of 12 months.

Programming Language	2020	2015	2010	2005	2000	1995	1990	1985
Java	1	2	1	2	3	-	-	-
C	2	1	2	1	1	2	1	1
Python	3	7	6	6	22	21	-	-
C++	4	4	4	3	2	1	2	12
C#	5	5	5	8	8	-	-	-
Visual Basic .NET	6	10	-	-	-	-	-	-
JavaScript	7	8	8	9	6	-	-	-
PHP	8	6	3	4	27	-	-	-
SQL	9	-	-	97	-	-	-	-
Objective-C	10	3	21	37	-	-	-	-
Lisp	31	18	16	13	14	5	3	2

2

Porque estudar conceitos de LPs

Capacidade de expressar ideias

- A profundidade com a qual pessoas podem pensar é influenciada pelo poder de expressividade da linguagem que elas usam.
- É difícil criar conceitos de estruturas que não podemos descrever verbalmente ou expressar na escrita
- Isso se aplica a programadores, no desenvolvimento de softwares
- Conhecer uma variedade mais ampla de recursos de LPs podem aumentar a faixa de processos mentais para solução de problemas

Embasamento para escolher

- Quando precisamos escolher uma LP para um novo problema, tendemos a escolher aquela em que temos mais familiaridade
- Se conhecemos mais LPs, temos mais capacidade de escolher uma LP que inclua os recursos mais adaptados às características do novo problema

Habilidade para aprender novas LPs

- O entendimento preciso dos conceitos de LPs facilita entender como tais conceitos são incorporados em uma LP específica
  - Conhecer conceitos de OO facilita aprender Java, pro exemplo

Importância da implementação

- Conhecer detalhes da implementação da LPs permite entender melhor como os programas serão executados em determinadas LPs
  - Entender a complexidade de chamadas a subprogramas evita que um programadores criem programas ineficientes por chamar constantemente um subprograma muito pequeno

Melhor uso da LP

- Muitas LPs recentes são projetos grandes e complexos
- Quanto mais recursos conhecidos, mais os programadores podem tonar seus programas eficientes

Avanço geral da computação

- Pelo histórico das LPs (aula anterior), nem sempre a LP mais popular era a melhor disponível
  - Se o ALGOL 60 fosse mais usado que o Fortran em 1960...
- Conhecer uma variedade maior de LPs permite que as melhores LPs sejam escolhidas e influenciem o desenvolvimento de novas linguagens

3

Domínios de Programação

Científico

- 1940: primeiros computadores digitais
  - Inventados e usados para aplicações científicas
  - Estruturas de dados simples
  - Computações aritméticas com ponto flutuante
  - Preocupação com desempenho de execução
- Fortran é o exemplo mais representativo

Empresarial

- O uso de computadores digitais para aplicações comerciais iniciou em 1950
  - COBOL, apareceu em 1960 e ainda continua a ser utilizado em muitos sistemas legados
- Linguagens para domínio empresarial devem
  - Ter facilidade para gerar relatórios elaborados
  - Descrever e armazenar números decimais (ponto fixo) e caracteres

Inteligência Artificial

- Caracterizada pela computação simbólico, ao invés de apenas numérica
  - Computação simbólica: manipulação de dados nominais
  - Manipulação de listas (facilita a manipulação de dados nominais)
- LISP foi a primeira LP para aplicações de IA (1959)
- Atualmente, utiliza-se várias outras LPs: Python, Java, Prolog

Software de sistemas

- Sistemas operacional e ferramentas de suporte à programação de outros sistemas
  - Utilizados quase continuamente
  - Devem ser muito eficientes
  - Recursos de baixo nível que permitam que aplicativos comuniquem-se com dispositivos externos
- Sistema UNIX é escrito quase totalmente em C

Programação Web

- A WWW é mantida por uma coleção bem diversificada com linguagens de marcação e linguagens de programação.
- Necessidade de apresentação dinâmica de conteúdos
  - Linguagens híbridas
  - Códigos de programação embarcados em documentos XHTML (Javascript, PHP)

4

Como avaliar uma LP

Como recursos de uma LP impacta no desenvolvimento de software

Critérios de avaliação

- Características que impactam nos critérios

	Critério		
Característica	**Legibilidade**	**Facilidade de Escrita**	**Confiabilidade**
Simplicidade	✔	✔	✔
Ortogonalidade	✔	✔	✔
Tipos de Dados	✔	✔	✔
Projeto de Sintaxe	✔	✔	✔
Suporte para Abstração		✔	✔
Expressividade		✔	✔
Checagem de Tipos			✔
Controle de Exceção			✔
Renomeação Restrita			✔

Critério: Legibilidade

- Facilidade com que programas podem ser lidos e entendidos
- Antes de 1970
  - LPs pensadas mais na máquina que nos programadores
- Depois de 1970 a codificação devia atender ao conceito de ciclo de vida de software
  - Manutenção determinada pela legibilidade do programa

Critério: Facilidade de Escrita

- Quão facilmente uma LP pode ser usada para criar programas para um domínio
- A maior parte das características que afetam a Legibilidade também afeta a Facilidade de Escrita
  - *O processo de escrita de um programa requer a releitura da parte já escrita*

Critério: Facilidade de Escrita

- Como a Legibilidade, depende do domínio do problema a ser resolvido
  - Facilidade de escrita de C e Visual Basic são drasticamente diferentes para criar uma interface gráfica para uso do programa
    - Visual Basic foi feito para isso, C, não

Critério: Confiabilidade

- Um programa é dito confiável quando está de acordo com suas especificações, em todas as condições a que seja submetido

Característica: Simplicidade

- Uma LP com muitas construções básicas é mais difícil de aprender do que uma com poucas
- Uma linguagem complexa, com muitas construções, acaba tendo apenas um subconjunto de suas construções utilizada por programados
  - Esses subconjuntos acabam por ser diferentes para cada programador
  - Assim, um programador pode ter dificuldade de ler o programa do outro

Característica: Simplicidade

- Outro complicador é a multiplicidade de recursos
- Java
  - `count = count + 1`
  - `count += 1`
  - `count++`
  - `++count`
- *As duas últimas sentenças são um pouco diferentes mas todas têm o mesmo significado quando usadas em expressões isoladas*

Característica: Simplicidade

- Outro complicador é a sobrecarga de operadores, na qual um operador tem mais de um significado
- Inclusive, quando é permitido ao programador definir um novo significado a um operador
  - *O operador + é intuitivo somar inteiros e ponto flutuante, mas e concatenar listas e strings?*

Característica: Ortogonalidade

- Indica que um conjunto relativamente pequeno de construções primitivas (básicas) de uma LP pode ser combinado para construir outras estruturas, inclusive de controle e de dados
- Está fortemente relacionada à simplicidade
  - Quanto mais ortogonal um projeto, menor o número necessário de exceções às regras da LP

Característica: Ortogonalidade

- Exemplos de falta de ortogonalidade:
  - C tem duas formas de tipos de dados estruturados: vetores e registros (structs)
    - Registros podem ser retornados por funções
    - Vetores, não
  - Um membro de uma estrutura pode ser qualquer tipo de dados, menos void ou uma estrutura do mesmo tipo
  - Parâmetros são passados por valor, a menos que sejam vetores

Característica: Tipos de dados

- Mecanismo para definir tipos e estruturas de dados que permitam ao programados resolver mais facilmente um problema
- Em C, não temos o tipo Booleano
  - `timeOut =1` (válido em C, mas ilegível)
  - `timeOut = true` (legível, mas não existe em C)

Característica: Projeto de Sintaxe

- A sintaxe, ou forma, das construções em uma LP afetam a Legibilidade e Facilidade de Escrita
- Por exemplo:
  - Formato dos identificadores
    - tamanhos curtos ou longos
    - caracteres permitidos
  - Palavras reservadas
- Em geral, um bom projeto de sintaxe deve projetar comandos que reflitam, em sua aparência, o seu propósito

Característica: Suporte a abstração

- Habilidade de definir e usar estruturas ou operações complicadas para permitir que muitos detalhes de um problema sejam ignorados (desde que não afete a solução do mesmo)
- Abstração de processos e de dados

Característica: Suporte a abstração

- Abstração de processos
  - Uso de subprogramas para compor uma solução maior
  - Sem subprograma, um trecho de código teria de ser replicado em diversas partes da solução
- Abstração de dados
  - Considere árvore binária implementada em uma linguagem que não oferece ponteiros e nem gerenciamento de memória dinâmica?

Característica: Expressividade

- Pode ter diversos significados
- Em geral, uma LP expressiva especifica computações com um programa mais enxuto

Característica: Checagem de tipos

- Execução de testes para detectar erros de tipos de dados em um programa, tanto em fase de compilação quanto de execução
- A verificação de tipos em tempo de execução é cara
  - Em tempo de execução é desejável
  - Em Java, praticamente todas variáveis e expressões são verificadas em tempo de compilação e isso diminui drasticamente os erros de tipos em tempo de execução
- E nas linguagens não tipadas? O que pode ocorrer?

Característica: Controle de exceção

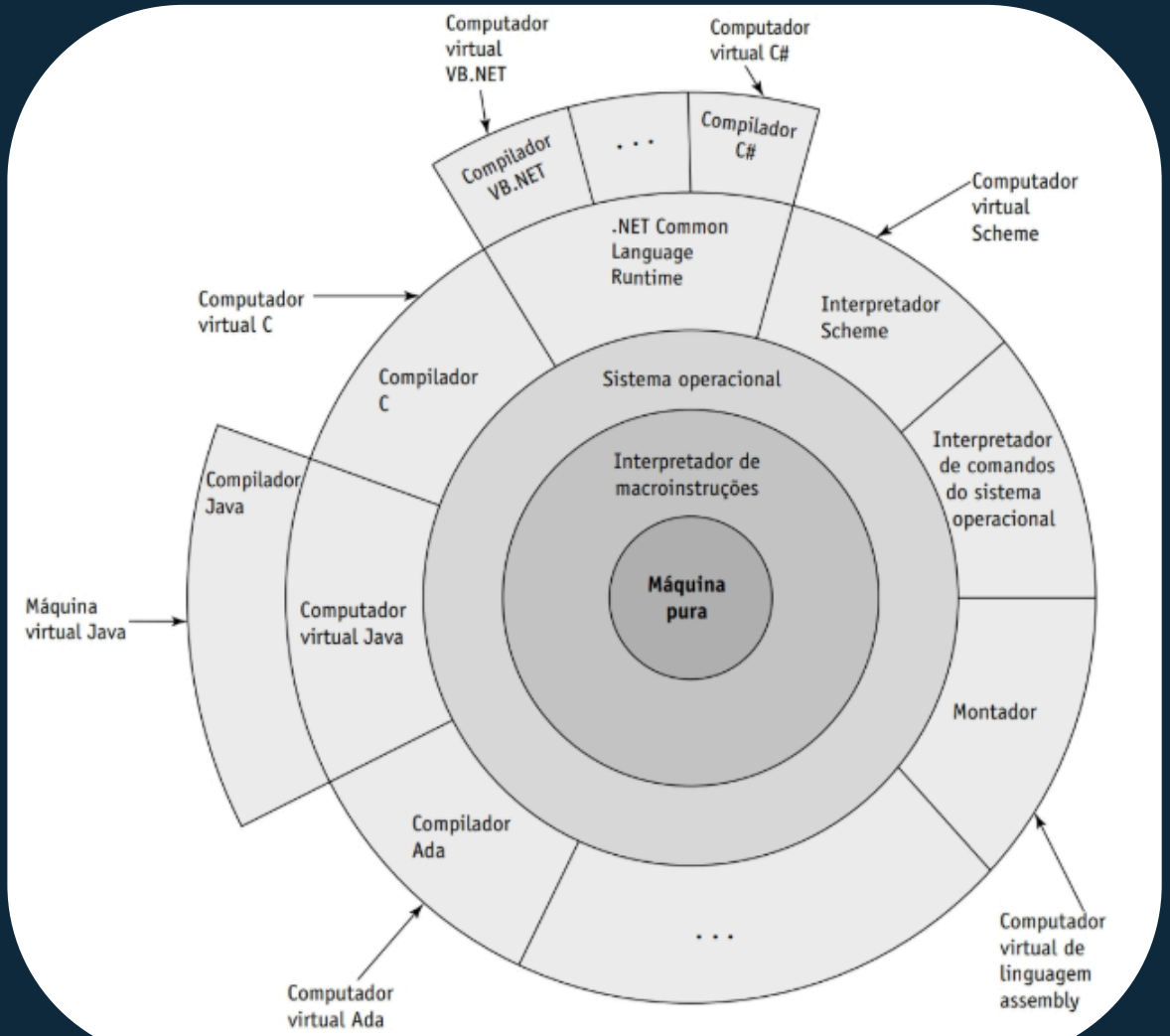
- Habilidade de um programa interceptar erros em tempo de compilação, tratá-los e continuar a execução do programa.
- As linguagens mais recentes são dotadas de estruturas de controle de exceção
  - Java, C++, Python
  - C e Fortran não têm tais estruturas

Característica: Renomeação restrita

- Renomeação ou utilização de apelidos é a possibilidade de ter um ou mais nomes para a mesma posição de memória
  - Por exemplo, dois ponteiros apontando para a mesma variável
- Em algumas linguagens o uso de apelidos serve para resolver deficiências nos recursos de abstrações de dados

5 Métodos de Implementação

5

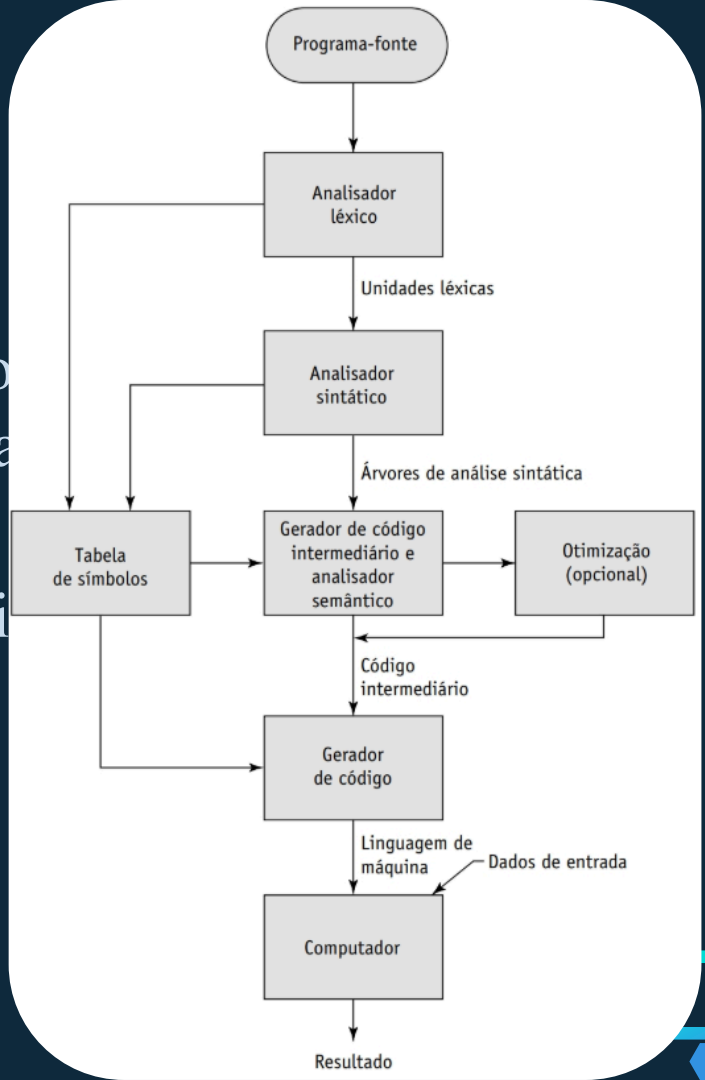

Computador virtual VB.NET
Computador virtual C#
Compilador VB.NET
. . .
Compilador C#
Computador virtual Scheme
.NET Common Language Runtime
Computador virtual C
Interpretador Scheme
Compilador C
Sistema operacional
Interpretador de comandos do sistema operacional
Compilador Java
Interpretador de macroinstruções
Máquina virtual Java
Máquina pura
Computador virtual Java
Montador
Compilador Ada
. . .
Computador virtual de linguagem assembly
Computador virtual Ada

Compilação

- Ocorre quando um código-fonte é traduzido para linguagem de máquina, a qual pode ser executada diretamente no computador
- O programa compilado (código de máquina) tem execução muito rápida
  - C, COBOL, C++

Compilação

- Ocorre quando um código-fo
  linguagem de máquina, a qua
  diretamente no computador
- O programa compilado (códi
  execução muito rápida
  - C, COBOL, C++



Interpretação Pura

- Oposto da compilação
- O código-fonte é ”executado” por outro programa, o interpretador
  - ciclo de obtenção-execução para cada sentença (comando) do programa (código-fonte)
- Execução e, em geral, 10 a 100 vezes mais lento que no processo com compilação
  - Python, Javascript, PHP

Implementação híbrida

- Meio termo entre compilação e interpretação pura
- O programa é traduzido para linguagens intermediárias, projetadas para facilitar a interpretação
  - Perl, Java
- São mais rápidos que interpretação pura
- Uma variação são os sistemas JIT (Just-in-Time)
  - Usam linguagem intermediária
  - Compilam partes do código-fonte para linguagem de máquina e usam para chamadas posteriores

6 Ambientes de Programação

Ambientes de Programação

- Coleção de ferramentas utilizadas no desenvolvimento de software
- Simplificadamente
  - Sistemas de arquivos, editor de texto, ligador e compilador
- UNIX é um dos ambientes mais antigos (desde 1970)
- Atualmente conta com interface gráfica com usuário

IDEs

- Grande coleção de ferramentas para assessorar o desenvolvimento
- Interface de usuário uniforme
  - Visual Studio .NET
  - NetBeans
  - Eclipse

  - Conhece algum outro?

Referência Bibliográfica

Sebesta, R. W. (2011). *Conceitos de Linguagens de Programação*. 9 ed. Bookman.

Capítulo 1