# Linguagem de Programação e Compiladores

Fernando Antônio Asevedo Nóbrega

Instituto de Ciências Matemáticas e de Computação – USP

SCC-206 Introdução à Compilação
24 de abril de 2012

# Sumário

1. Introdução
2. Características
   - Domínio ou Propósito
   - Variáveis, Nomes e Tipo
3. Paradigmas
   - Funcional
   - Lógico
   - Procedural
   - Orientado à Objetos
   - Multi-paradigma
4. Linguagem LUA
   - Características Lua

Quais, e como, características de Linguagens de Programação interferem no desenvolvimento de um Compilador

# Domínio

**Em qual área será utilizada a linguagem?**

- Científico
  - Fortran, Matlab®, Ocatve, Ada
- Comercial
  - Cobol
- Geral
- Inteligência Artificial
  - LISP, Prolog, Scheme
- Móvel
  - Java
- *Scripting*
  - Lua, Perl, Python
- Web
  - JavaScript, PHP
- Específicas*
  - RPG (Relatórios)

**Isso interfere na construção de um Compilador?**

**Isso é léxico, sintático ou semântico?**

# Tipagem de Variáveis

- Declaração: Explícita X Implícita
  - C, Java X Python, PHP
- Tipagem: Dinâmica X Estática
  - PHP, Pyton X C, Java
- Tipagem: Fraca X Forte
  - C, PHP X Java

Isso é léxico, sintático ou semântico?

# Verificação de Tipo e Exceções

Verificar tipos das variáveis e possíveis exceções, como acessar uma posição não existente de um vetor

- C: não trata posição inválida de vetores
- Java: trata uso de posições inválidas
- : Verificação em compilação x execução

Léxico, Sintático ou Semântico?

O que muda em um compilador que possui esses recursos para outro que não possui?

# Como é a LALG

| | |
|---|---|
| Domínio | |
| Declaração | |
| Tipagem | |
| Exceções | |
| Paradigma | |

# Como é a LALG

| Domínio | Geral (acadêmico) |
|---|---|
| Declaração | |
| Tipagem | |
| Exceções | |
| Paradigma | |

# Como é a LALG

| | |
|---|---|
| Domínio | Geral (acadêmico) |
| Declaração | Explícita |
| Tipagem | |
| Exceções | |
| Paradigma | |

# Como é a LALG

| | |
|---|---|
| Domínio | Geral (acadêmico) |
| Declaração | Explícita |
| Tipagem | Estática e Fraca |
| Exceções | |
| Paradigma | |

# Como é a LALG

| | |
|---|---|
| Domínio | Geral (acadêmico) |
| Declaração | Explícita |
| Tipagem | Estática e Fraca |
| Exceções | Não trata |
| Paradigma | |

# Como é a LALG

| | |
|---|---|
| Domínio | Geral (acadêmico) |
| Declaração | Explícita |
| Tipagem | Estática e Fraca |
| Exceções | Não trata |
| Paradigma | Procedural |

# O que são paradigmas?

- Formas diferentes de resolver um problema
- Adequações para algumas áreas
- Estudos formais de computabilidade (Paradigma Funcional)
- Por que é importante reconhecer paradigmas?

# O que são paradigmas?

- Formas diferentes de resolver um problema
- Adequações para algumas áreas
- Estudos formais de computabilidade (Paradigma Funcional)
- Por que é importante reconhecer paradigmas?

# Funcional

- Baseada em funções matemáticas
- Não há comandos de iteração, devem ser convertidos para funções recursivas
- Muito útil para manipulação de símbolos (IA)
- Desenvolvidas para estudo de computabilidade
  - LISP
- *Como a matemática (funções) resolve um problema?*

Lisp, Scheme, Haskell

# Lógico

- Sintaxe baseado em lógica matemática (Clausulas lógicas)
- Cálculo de Predicados
- Não sequencial
- Aplicam, em geral, provadores de teoremas
- Aplicado em IA
- Não possui comandos de iteração
- *O que deve ser representado para resolver um problema?*

Prolog (e variações)

# Procedural

- Sequencia de comandos
- Paradigma mais comum aos programadores
- Possuem comandos de iteração e desvio
- *Qual a sequencia de passos para resolver um problema?*

C, C++(meio termo), Basic

# Orientado à Objetos

- Não necessariamente uma sequencia de comandos
- Objetos, que representam entidades, trocam mensagens
- Herança
- Abstração dos dados
- *Como entidades relacionam-se para resolver um problema?*

C++(meio termo), Java, Python, Perl, Ruby

# Multi-paradigma

- Agregam características de mais do que um paradigma
- Em geral, a maioria das linguagem possuem um paradigma dominante e agregam características de outros paradigmas
  - Funcional: Scheme
  - OO: Java, Ruby, Perl
- Existem linguagens projetadas para multi-paradigmas

Oz, Scala[a]

[a] Usada pelo Twitter

# Linguagem LUA

- Desenvolvida na PUC-RIO – Brasil
  - download: http://www.lua.org/download.html
- Atualmente, mantida pelo LabLua
  - http://www.lua.inf.puc-rio.br
- Famosa pelo uso em games
- Ironicamente, mais conhecida fora do Brasil
- Uso de tabelas associativa (pode-se entender como hash)
- Quem usa:

# Léxico da Lua

- Identificador
  - Qualquer cadeia de letras, dígitos e sublinhados que não inciam com dígito
  - Sensível ao caso
- Palavras reservadas
  - and, break, do, else, elseif, end, false, for, function, if, in, local, nil, not, or, repeat, return, then, true, until, while
- Símbolos permitidos
  - ^ + − ∗/%# == =<=>=<>= ( ) { } [ ] ; : , . .. ...

# Precedência de Operadores

| | |
|---|---|
| Maior | ^ |
| | **not** # - (unário) |
| | * / % |
| | + − |
| | .. |
| | < > <= >= = == |
| | **and** |
| Menor | **or** |

# Estruturas de Controle

comando ::= **while** exp **do** bloco **end**

comando ::= **repeat** bloco **until** exp

comando ::= **if** exp **then** bloco {**elseif** exp **then** bloco} [**else** bloco] **end**

comando ::= **for** nome **=** exp **,** exp [**,** exp] **do** bloco **end**

comando ::= **for** listadenomes **in** listaexp **do** bloco **end**

# Retorno e Escopo

## Retorno

- **comando** ::= **return** [ listaexp ]
  - O que isso sugere?

## Escopo

Lua possui escopo léxico, onde um identificador é visível a partir do momento que foi declarado até o fechamento (**end**) do bloco mais interno.

# Retorno e Escopo

## Retorno

- **comando** ::= **return** [ listaexp ]
  - O que isso sugere?
  - Uma função pode retornar mais de um valor

## Escopo

Lua possui escopo léxico, onde um identificador é visível a partir do momento que foi declarado até o fechamento (**end**) do bloco mais interno.

# Retorno e Escopo

## Retorno

- **comando** ::= **return** [ listaexp ]
  - O que isso sugere?
  - Uma função pode retornar mais de um valor

## Escopo

Lua possui escopo léxico, onde um identificador é visível a partir do momento que foi declarado até o fechamento (**end**) do bloco mais interno.

# Características Lua

| | |
|---|---|
| Compilação | |
| Domínio | |
| Declaração | |
| Tipagem | |
| Exceções | |

# Características Lua

| Compilação | Interpretada |
|---|---|
| Domínio | |
| Declaração | |
| Tipagem | |
| Exceções | |

# Características Lua

| | |
|---|---|
| Compilação | Interpretada |
| Domínio | *Scripting* |
| Declaração | |
| Tipagem | |
| Exceções | |

# Características Lua

| | |
|---|---|
| Compilação | Interpretada |
| Domínio | *Scripting* |
| Declaração | Implícita |
| Tipagem | |
| Exceções | |

# Características Lua

| | |
|---|---|
| Compilação | Interpretada |
| Domínio | *Scripting* |
| Declaração | Implícita |
| Tipagem | Dinâmica |
| Exceções | |

# Características Lua

| | |
|---|---|
| Compilação | Interpretada |
| Domínio | *Scripting* |
| Declaração | Implícita |
| Tipagem | Dinâmica |
| Exceções | trata |

# Referências


- Sebesta, Robert W.(2009) Concepts of programming languages. 9 ed. Editora Pearson
- Ierusalimschy, R.; de Figueiredo, L. H.; Waldemar Celes Manual de Referência de Lua 5.1. http://www.lua.org/manual/5.1/pt/manual.html