Universidade do Estado do Rio Grande do Norte – UERN
Campus Natal – Ciência da Computação
Disciplina: Compiladores (Rosiery)

# ANALISADOR LÉXICO

Antonio Porto, Jônata Marcelino, Mariêta Cunha

# Definição

- Análise léxica é o processo de analisar a entrada de linhas de caracteres e produzir uma seqüência de símbolos chamado "símbolos léxicos" (tokens).
- A Análise Léxica é a forma de verificar determinado alfabeto. Quando analisamos uma palavra, podemos definir através da análise léxica se existe ou não algum carácter que não faz parte do nosso alfabeto.

# Praticando a Análise Léxica com Lua

- Informações Básicas
  - Lua é uma linguagem de programação rápida e leve, projetada para estender aplicações. Lua é inteiramente projetada, implementada e desenvolvida no Brasil. Ela foi criada por uma equipe na PUC-Rio (Pontifícia Universidade Católica do Rio de Janeiro).

# Praticando a Análise Léxica com Lua

- Características da Linguagem
  - Lua é  uma linguagem tipada dinamicamente, é interpretada a partir de bytecodes para uma máquina virtual.
    - Essas características fazem de Lua uma linguagem ideal para configuração, automação (scripting) e prototipagem rápida.

# Escopo da Linguagem

- Estrutura Condicional

```
c = 10 and 20
print(c)
```

- Estrutura de Repetição

```
a = 0
repeat
    print ( a )
    a = a + 1
until a == 5
```

# Alfabeto de Lua

- {a...z, A...Z, 0...9, .. ,(, ), =, “ ,<, >, <=, >=, ==, ~=, -, +, *, . , /, [, ], -- }

# Definição das Expressões Regulares

- **Identificador:**
  `^[^\p{Digit}\p{Punct}áàäâÁÀÄÂéèëêÉÈËÊíìïîÍÌ ÏÎóòöôÓÒÖÔúùüûÚÙÜÛ](\w)*$`
- **Funções:**
  `^(io\\.write|return|print|function)$`
- **Operadores Lógicos:** `^(and|or|not)$`
- **Operador Aritmético:** `^[-|+|\*|/]$`
- **Símbolos:** `^(=|,)$`
- **Operador relacional:** `^(<|>|<=|>=|==|~=)$`
- **Delimitadores:** `^(end|\(|\)|\")$`
- **Laços iterativos:** `^(while|do|repeat|until)$`
- **Nil:** `^(nil)$`

# Definição das Expressões Regulares

- **Tomada de decisão:** `^(if|then|else|elseif)$`
- **Number:** `^(?![+-]?\.)(?!\$)(?!\p{Alpha})(?!\p{Punct}&&[^+-])(?!\p{Space})(?![\p{Graph}&&[^\d\+\-]])(((?![+-]0)(\+|-)?[^0]+[0-9]*+(\.[0-9]*))|((?![+-]0)(\+|-)?^[^0][0-9]*)|((\+|-)?0?+(\.[0-9]+))|0)?$`
- **Comentário:** `^-{2,}([\p{Punct}]|\s|\w|\d|\p{L})*$`
- **Concatenação:** `^(\.\.)$`
- **String:** `^\"(\p{Digit}|\p{Alpha}|\s|[\p{Punct}&&[^\"]]|[^\p{Z}\p{C}&&[^\"]])*\"$`

# Definição das Expressões Regulares

- **Tomada de decisão:** `if|then|else|elseif`
- **Number:** `^(?=.+)(?:[1-9]\\d*|0)?(?:\\.\\d+)?$`
- **Comentário:** `--`
- **Concatenação:** `. .`
- **String:** `^\"[a-zA-Z0-9áãàêéíóôöőúüűÁÉÍÔÓÖŐÚÜŰ\\s=-]*\"$`
- **Comentários:** `-{1,2}[a-zA-Z0-9áãàêéíóôöoúüuÁÉÍÔÓÖOÚÜU'.\"=-_*\\s]*`

# Demonstração Prática

- Para a visualização prática acesse o seguinte endereço: https://github.com/johny83/compiladorlua

# Referências


- TUTORIAL. A Linguagem de Programação Lua. Disponível em <http://www.lua.org/portugues.html>

- REINERT, Karoline. Linguagem de Programação Lua. Artigo. Disponível em: <http://www.comp.ita.br/~gian/tes04/trabalhos/lua-all.signed.pdf>

- Java Regex Tester. Disponível em <http://java-regex-tester.appspot.com>

- Java Regex Tutorial. Disponível em <http://www.vogella.com/articles/JavaRegularExpressions/article.html>