# Curso: Engenharia de Computação

## Linguagens Formais e Compiladores

Prof. Clayton J A Silva, MSc
clayton.silva@professores.ibmec.edu.br

ibmec

# BNF e análise léxica

- **A definição da linguagem deve determinar a interpretação** a ser observada na análise – deve fornecer, por exemplo, as regras de eliminação de ambiguidade

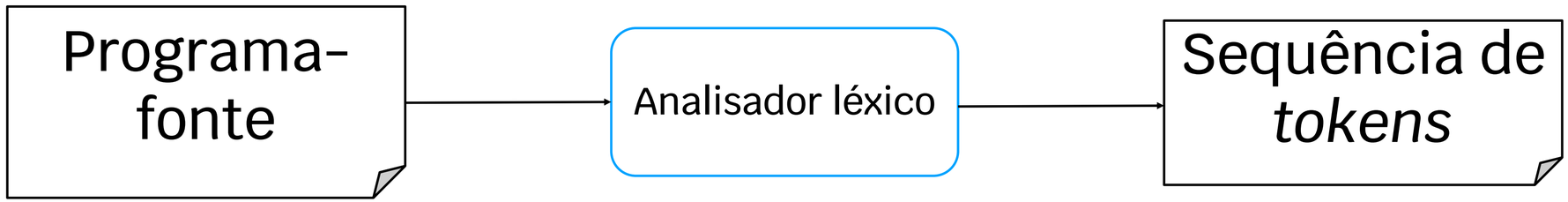
- **A descrição da sintaxe da linguagem** pode ser realizada utilizando-se a forma de Backus-Naur ou BNF

- A perspectiva que discutiremos se baseia nas premissas: **existe a descrição sintática da linguagem**, **existe um código escrito por um programador** – como analisar se o código respeita a sintaxe?

# Varredura ou análise léxica

# Varredura ou análise léxica

- É a fase de um compilador que **lê o arquivo de um programa-fonte como um arquivo de caracteres** e **o converte em uma sequência de *tokens***.



# Varredura ou análise léxica

- Genericamente, os *tokens* podem ser **palavras reservadas ou palavras-chave**, **identificadores**, **símbolos especiais** e **literais ou constantes**
- Os *tokens* classificam os *lexemas*; existem *tokens* que podem classificar somente um lexema; existem tokens que podem classificar infinitos lexemas
- Qualquer **valor** associado a um *token* recebe o nome de um **atributo** do *token*
- O sistema de varredura poderá definir vários atributos a cada *token* quanto necessários para prosseguir o processamento

# Varredura ou análise léxica

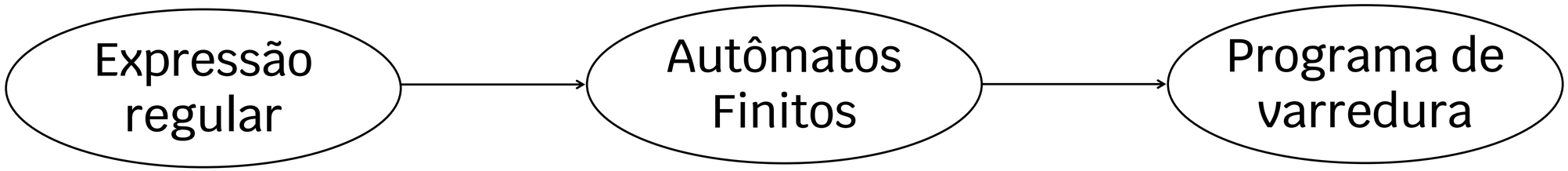
- O **conjunto de atributos** de um token pode ser atribuído a uma estrutura de dados chamada **registro** do token

```
struct
  { TokenTipo tokenval;
    tipo atributo1token;
    ...
  }
```

- O analisador sintático – que controla a análise léxica - normalmente chama o *token* por uma **função, que retorna o próximo *token* identificado no código**

# Varredura ou análise léxica

- O processo para desenvolver o programa de varredura (análise léxica) geralmente consiste em aplicar **métodos de especificação e de reconhecimento de padrões**: **expressões regulares** e **autômatos finitos**.



# Varredura ou análise léxica

- As expressões regulares constituem uma **notação padrão para representar padrões em cadeias de caracteres** que formam a estrutura léxica de uma LP

- As máquinas de estados finitos, ou autômatos finitos, são **algoritmos para reconhecimento de padrões** em cadeias de caracteres dadas por expressões regulares.

# Expressões regulares

- Representam padrões de cadeias de caracteres
- Uma expressão ***r*** é definida pelo **conjunto de cadeias de caracteres** com as quais ela casa, chamado de **linguagem gerada pela expressão regular**
- A expressão é definida por: $L(r) = \Omega$, onde $\Omega$ é o conjunto de cadeias de caracteres definido entre chaves
- Uma expressão regular ***r*** pode conter caracteres com significados especiais, chamados de **metacaracteres** ou **meta-símbolos**

# Expressões regulares básicas

1. Caracteres em separado do **alfabeto** ($\Sigma$), que casam com eles mesmos. Indica-se *L(**a**) = {a}*, onde **a** é a expressão regular e *a* é o símbolo do alfabeto.

2. C**adeia vazia**, ou seja, a cadeia sem caracteres, designada $\varepsilon$. Indica-se $L(\varepsilon) = \{\varepsilon\}$.

- **Conjunto vazio,** aquele para o símbolo que não casa com nenhuma cadeia de caracteres. Indica-se $L(\Phi) = \{\,\}$

# Operações de expressões regulares

(1) **Escolha entre alternativas**, representada pelo meta-símbolo |

(2) **Concatenação**, representada pela justaposição – sem meta-símbolos

(3) **Repetição** ou **fecho**, representada pelo meta-símbolo *

# Escolha entre alternativas (I)

- Se ***r*** e ***s*** são expressões regulares, então ***r I s*** é uma expressão regular que casa com qualquer cadeia que casa com ***r*** ou casa com ***s***.
- A operação pode ser definida por: $L(r|s) = L(r) \cup L(s)$
- Aplica-se a escolha com a cadeia vazia: $L(r|\varepsilon) = \{r, \varepsilon\}$

# Concatenação

- Se ***r*** e ***s*** são expressões regulares, então ***rs*** é uma expressão regular que casa com qualquer cadeia que casa com r concatenado com s.

- A operação pode ser definida por: $L(rs) = L(r)L(s)$

# Repetição ou fecho (*)

- Se ***r*** é expressão regular, então ***r**** é uma expressão regular que casa com qualquer concatenação finita de cadeias de caracteres, desde que cada cadeia case com ***r***.

- A operação pode ser definida por $L(r*) = L(r)*$

# Precedências e parênteses

- **Precedência**: 1. repetição, 2. concatenação, 3. escolha
- A precedência entre expressões regulares **pode ser definida pelo uso de parênteses**
- **Nomes** para expressões regulares: são usados para simplificar expressões longas – o nome é um meta-símbolo

# Extensões de expressões regulares

- **Uma ou mais repetições** – sem 0 repetição – Seja ***r*** uma expressão regular, ***r+*** representa uma ou mais repetições

- **Casamento com qualquer caractere do alfabeto** – Seja ***r*** uma expressão regular, ***r.*** representa que ***r*** pode ser concatenada com qualquer caractere

- **Intervalo de caracteres** – Pode-se usar **colchetes e um hífen** – a notação deve depender da **ordenação** do conjunto de caracteres a representar

# Extensões de expressões regulares

- **Caractere fora de um conjunto** – Pode-se utilizar um meta-símbolo para indicar um ‘não’ ou complemento. **Um meta-símbolo usado é o ~.**

- **Partes opcionais** – Pode-se utilizar a parte opcional como um caractere que concatena com a expressão regular. Para evidenciar que são opcionais, utiliza-se o **meta-símbolo ?** após a parte opcional

# Expressões regulares típicas para *tokens*

- **Números**. Podem ser apenas sequências de dígitos (números naturais), números decimais ou números com um expoente

  *nat = [0-9]+*

  *sinalNat = (+|-)? nat*

  *numero = sinalNat (.nat)? (E sinalNat)?*

# Expressões regulares típicas para *tokens*

- **Palavras reservadas** e **identificadores**.

  *reservadas = if | while | do | for |...*

  *letra = [a-z A-Z]*

  *digito = [0-9]*

  *identificador = letra(letra | digito)**

# Expressões regulares típicas para *tokens*

- **Comentários**. Normalmente ignorados durante a varredura, que precisa identificá-los e descartá-los.

# Referências


- Louden, Kenneth C.; Compiladores princípios e práticas; Capítulos 2.1 e 2.2; THOMSON

IBMEC.BR

/IBMEC

IBMEC

@IBMEC_OFICIAL

@IBMEC

Ibmec