# Conceitos de Linguagens de Programação

## Aula 05 – Análise Léxica (Implementação)

Edirlei Soares de Lima
<edirlei@iprj.uerj.br>

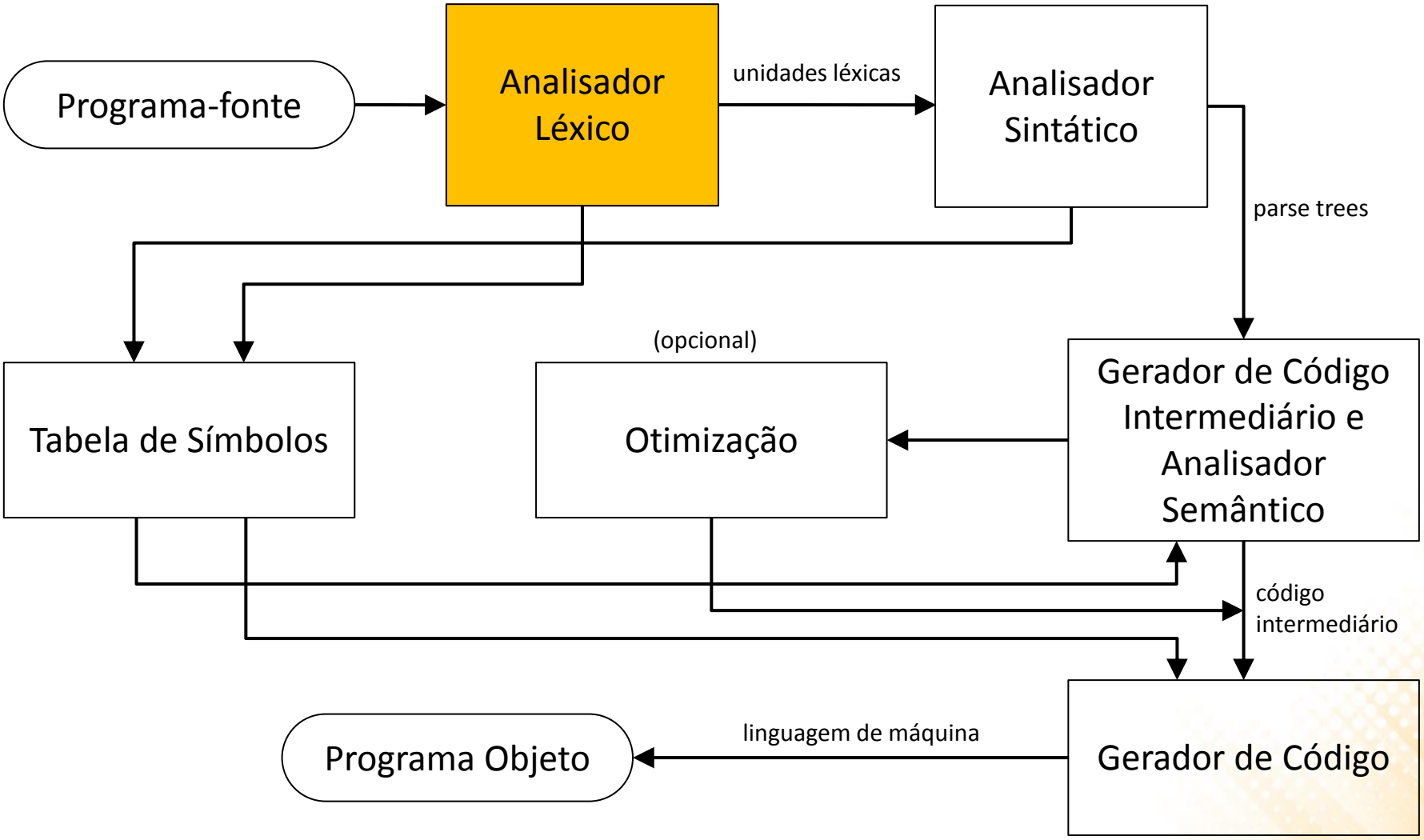
# Análise Sintática

- A maioria dos compiladores separam a tarefa da **análise sintática** em duas partes distintas:
  - Análise Léxica;
  - Análise Sintática;

- O **analisador léxico** trata as construções de pequena escala da linguagem (nomes, literais numérico, símbolos...);

- O **analisador sintático** trata as construções de larga escala da linguagem (instruções, expressões, unidades do programa...);

# Processo de Compilação



# Análise Léxica

- Motivos para separar a análise léxica da análise sintática:
  - **Simplicidade**: as técnicas para a análise léxica são menos complexas do que as exigidas para a análise sintática;
  - **Eficiência**: a separação facilita a otimização do analisador léxico;
  - **Portabilidade**: o analisador léxico lê arquivos de entrada contendo o código fonte do programa, sendo assim ele é de certa forma dependente da plataforma. Já o analisador sintático pode ser independente da plataforma.

# Análise Léxica

- Um analisador léxico é essencialmente um **verificador de padrões**.
- Um programa de entrada para um compilador é visto como uma **única cadeia de caracteres**.
- O analisador léxico **coleta os caracteres em agrupamentos lógicos** e atribuir códigos internos para os agrupamentos de acordo com a sua estrutura.

```
int gcd(int a, int b)
{
  while (a != b) {
    if (a > b) a -= b;
    else b -= a;
  }
  return a;
}
```

```
int | gcd | ( | int | a | , | int | b | ) | { | while | ( | a
!= | b | ) | { | if | ( | a | > | b | ) | a | -= | b | ; | else
b | -= | a | ; | } | return | a | ; | }
```

# Descrevendo a Sintaxe

- Os agrupamentos de caracteres são chamados de **lexemas** e os códigos internos são chamados de **tokens**.

- Exemplo:

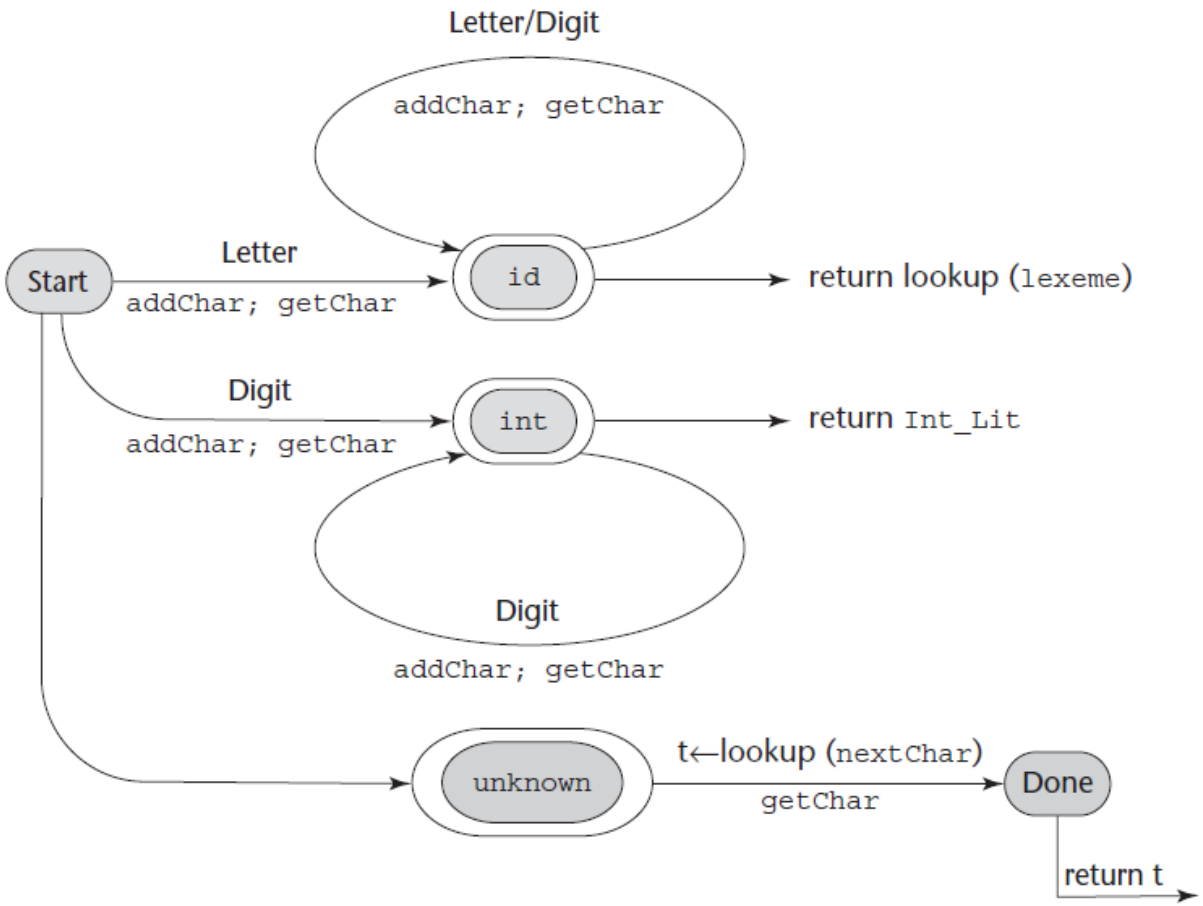
```
index = 2 * count + 17;
```

| **Lexemas** | **Tokens** |
|---|---|
| index | IDENTIFICADOR |
| = | OP_ATRIBUICAO |
| 2 | INT_LITERAL |
| * | OP_MULTIPLICAÇÂO |
| cont | IDENTIFICADOR |
| + | OP_SOMA |
| 17 | INT_LITERAL |
| ; | PONTO_E_VIRGULA |

# Análise Léxica

- Existem duas abordagens para se construir uma analisador léxico:
  - Escrever uma descrição formal dos padrões de símbolos da linguagem utilizando uma linguagem de descrição relacionada com **expressões regulares** e usar um software para gerar automaticamente o analisador. Exemplo: lex;
  - Construir **autômatos finitos** para descrever os padrões de símbolos da linguagem e escrever um programa para implementar e utilizar os autômatos.
    - Tabelas de transição ou diretamente em código.

# Implementação Direta em Código



# Implementação Direta em Código

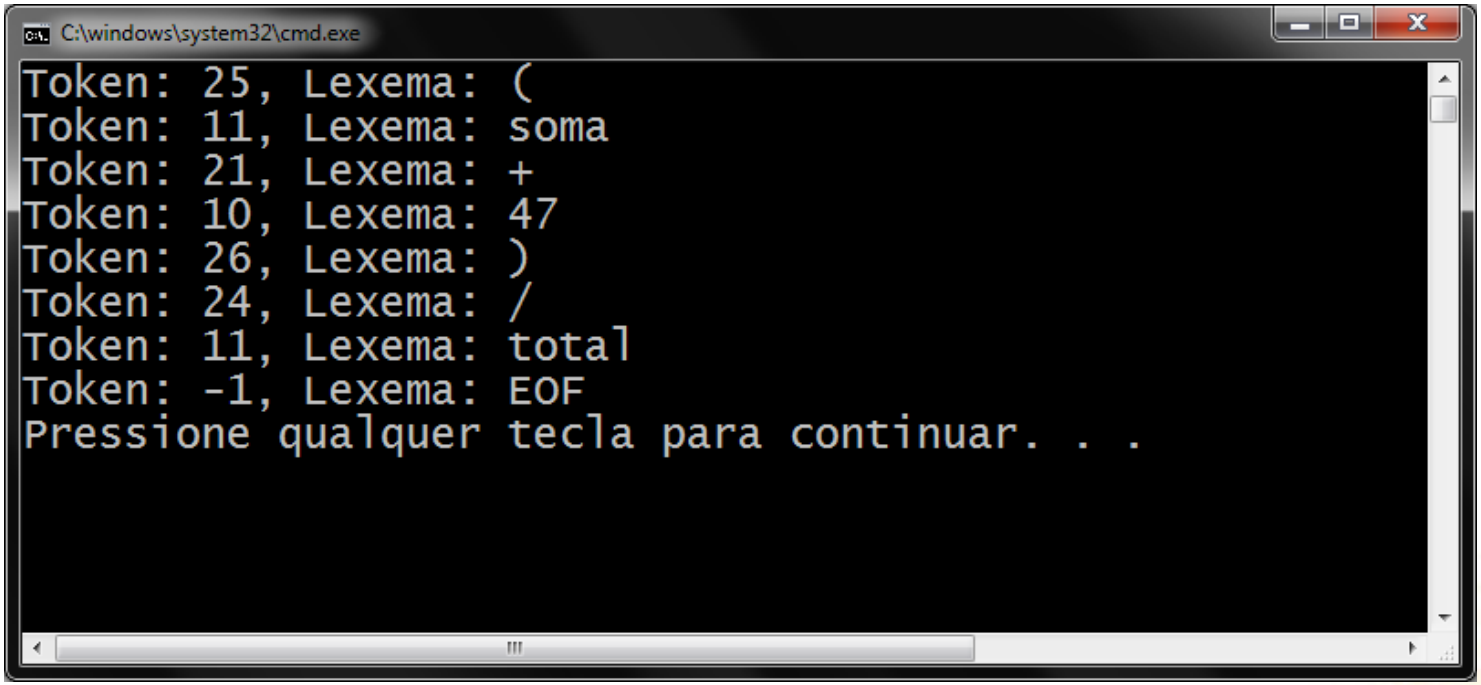
```
...
*nextChar = getNonBlank(code_file, *nextChar);
switch (nextChar->charclass)
{
  case LETTER:
    lexLen = addChar(lexeme, lexLen, nextChar->nextchar);
    *nextChar = getChar(code_file);
    while ((nextChar->charclass==LETTER)||(nextChar->charclass==DIGIT))
    {
      lexLen = addChar(lexeme, lexLen, nextChar->nextchar);
      *nextChar = getChar(code_file);
    }
    nextToken = IDENT;
    break;
  case DIGIT:
    lexLen = addChar(lexeme, lexLen, nextChar->nextchar);
    *nextChar = getChar(code_file);
    while (nextChar->charclass == DIGIT)
    {
      lexLen = addChar(lexeme, lexLen, nextChar->nextchar);
      *nextChar = getChar(code_file);
    }
    nextToken = INT_LIT;
    break;
    ...
```

http://www.inf.puc-rio.br/~elima/clp/projeto_compilador_v1.zip

# Implementação Direta em Código

- Entrada:

```
(soma + 47) / total
```



# Exercícios

- Continue a implementação do analisador léxico, acrescentando as seguintes características:

1) Reconhecer e identificar números decimais (separados por ponto). Exemplo:

```
(soma + 87.2) / total
```

2) Reconhecer e identificar expressões de atribuição. Exemplo:

```
resultado = (soma + 87.2) / total
```

3) Reconhecer e identificar programas compostos por mais de uma expressão de atribuição (separadas por ponto e virgula). Exemplo:

```
a = 8 + 42.75;
b = a * 0.8;
c = (a - b) / 2;
```

# Leitura Complementar

- Sebesta, Robert W. **Conceitos de Linguagens de Programação**. Editora Bookman, 2011.

- **Capítulo 4: Análise Léxica e Sintática**