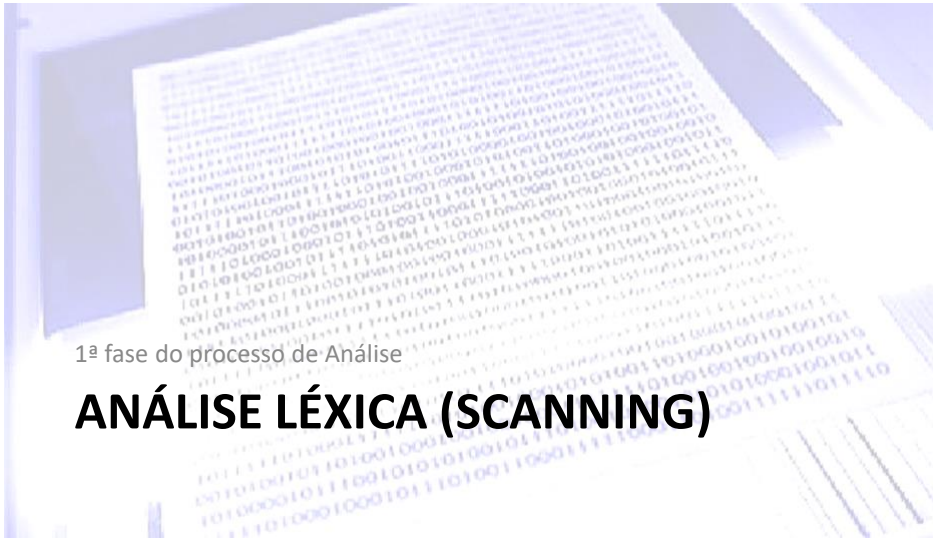




# Compiladores e Computabilidade

Prof. Leandro C. Fernandes  
UNIP – Universidade Paulista, 2018

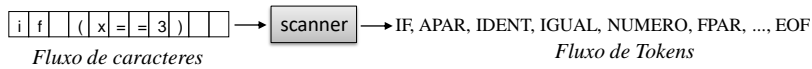


1ª fase do processo de Análise

## ANÁLISE LÉXICA (SCANNING)

## Tarefas de um Analisador Léxico

- Produzir símbolos terminais



- Ignorar e descartar símbolos irrelevantes

- espaços em branco
- caracteres de tabulação
- caracteres de controle (CR e LF)
- comentários

- Tokens possuem uma estrutura sintática

identif = letra {letra | dígito}

número = dígito {dígito}

if = "if" "if"

igual = "=" "="

...

Por quê o analisador léxico não é uma parte do analisador sintático?

3

## Por quê o analisador léxico não é uma parte do analisador sintático?

- Isso deixaria o analisador sintático mais complicado de ser construído
  - Ex. dificulta a distinção entre palavras reservadas e identificadores

```
Statement = ident "=" Expr ";"
           | "if" "(" Expr ")" ...
```

- Precisaria ser reescrito para a forma:

```
Statement = "if" ( "if" "(" Expr ")" ...
                | notF {letter | digit} "=" Expr ";"
                )
           | notI {letter | digit} "=" Expr ";"
```

- O scanning deve eliminar brancos, tabulações, fins de linha e comentários
  - Esses caracteres podem ocorrer em qualquer lugar do código, levando a gramáticas muito complexas

```
Statement = "if" {Blank} "(" {Blank} Expr {Blank} ")" {Blank} ...
Blank = " " | "\t" | "\n" | "\t" | Comment.
```

- Tokens podem ser descritos por linguagens regulares
  - mais simples e mais eficientes que as gramáticas livres de contexto

4

# Gramáticas Regulares

Definição:

- Um gramática é dita regular se puder ser descrita por produções na forma:

$A = a$        $a, b \in \text{Símbolos Terminais}$   
 $A = b B$      $A, B \in \text{Símbolos Não-terminais}$

- Ex: Gramática de nomes

$\text{Identif} = \text{letra}$   
 $\quad \quad | \text{letra Restante}$   
 $\text{Restante} = \text{letra}$   
 $\quad \quad | \text{dígito}$   
 $\quad \quad | \text{letra Restante}$   
 $\quad \quad | \text{dígito Restante}$

Ex: Derivação do nome xy3

$\text{Identif}$   
 $\quad \quad \square$   
 $\text{letra Restante}$   
 $\quad \quad \square$   
 $\text{letra letra Restante}$   
 $\quad \quad \square$   
 $\text{letra letra dígito}$

Definição alternativa

- Um gramática é chamada regular se puder ser descrita por meio de uma simples e não recursiva produção EBNF.

$\text{Ident} = \text{letra} \{ \text{letra} \mid \text{dígito} \}$

5

## Exemplos

- Podemos transformar a gramática abaixo numa gramática regular?

$E = T \{ "+" T \}$   
 $T = F \{ "*" F \}$   
 $F = \text{id}$

Depois de substituir F em T

$T = \text{id} \{ "*" \text{id} \}$

Depois de substituir T em E

$E = \text{id} \{ "*" \text{id} \} \{ "+" \text{id} \{ "*" \text{id} \} \}$

A gramática é regular

- Podemos transformar a gramática abaixo numa gramática regular?

$E = F \{ "*" F \}$   
 $F = \text{id} \mid "(" E ")"$

Depois de substituir F em E

$E = (\text{id} \mid "(" E ")") \{ "*" (\text{id} \mid "(" E ")") \}$

Substituir E em E não nos ajuda mais.

A recursão central não pode ser eliminada.

A gramática não é regular

6

## Limitações das Gramáticas Regulares

- Gramáticas regulares não podem lidar com estruturas aninhadas
  - Não são capazes de manipular recursão central!
- Porém esse tipo de construção é importante na maioria das linguagens de programação
 

|                      |                                                            |
|----------------------|------------------------------------------------------------|
| Expressões aninhadas | <code>Expr ::= "(" Expr ")" ...</code>                     |
| Comandos aninhados   | <code>Comando ::= "do" Comando "while" "(" Expr ")"</code> |
| Classes aninhadas    | <code>Classe ::= "class" "{" ... Classe ... "}"</code>     |

  - Para construções desse tipo precisamos de Gramáticas Livres de Contexto
- A maioria das estruturas léxicas são regulares
 

|                     |                                                 |
|---------------------|-------------------------------------------------|
| Nomes               | <code>letra { letra   dígito }</code>           |
| Números             | <code>dígito { letra   dígito }</code>          |
| Strings             | <code>"\" { qqCaractereExcetoAspas } "\"</code> |
| Palavras reservadas | <code>letra { letra }</code>                    |
| Operadores          | <code>"&gt;"   "="   "+"   ...</code>           |

Exceto comentários aninhados:

```
/* ... /* ... */ ... */
```

o scanner deve manipulá-los de uma maneira especial

7

## Expressões Regulares

- Notação alternativa para Gramáticas Regulares

### Definição

- $\varepsilon$  (cadeia vazia) é uma expressão regular
- um símbolo terminal é uma expressão regular
- Se  $\alpha$  e  $\beta$  são expressões regulares as expressões abaixo também são regulares:

|                  |                                                                    |
|------------------|--------------------------------------------------------------------|
| $\alpha\beta$    |                                                                    |
| $(\alpha \beta)$ | $\varepsilon   \alpha$                                             |
| $(\alpha)?$      | $\varepsilon   \alpha   \alpha\alpha   \alpha\alpha\alpha   \dots$ |
| $(\alpha)^*$     | $\alpha   \alpha\alpha   \alpha\alpha\alpha   \dots$               |
| $(\alpha)^+$     | $\alpha   \alpha\alpha   \alpha\alpha\alpha   \dots$               |

### Exemplos:

- |                                          |                      |
|------------------------------------------|----------------------|
| • <code>"w" "h" "i" "l" "e"</code>       | <code>while</code>   |
| • <code>letra { letra   dígito }*</code> | <code>nomes</code>   |
| • <code>dígito+</code>                   | <code>números</code> |

8

# Autômatos Finitos Determinísticos

- Podem ser usados para analisar linguagens regulares

Exemplo

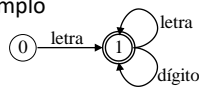


Tabela da **função de transição de estados**:

| d  | letra | dígito |                                                                         |
|----|-------|--------|-------------------------------------------------------------------------|
| s0 | s1    | erro   | "finito", porque d apresenta uma quantidade finita de estados possíveis |
| s1 | s1    | s1     |                                                                         |

Definição

- Um autômato finito determinístico é um quintupla  $(S, I, \delta, s_0, F)$

S            conjunto de estados  
I            alfabeto de entrada  
 $\delta: S \times I \rightarrow S$     função de transição  
 $s_0$         estado inicial  
F            conjunto de estados finais

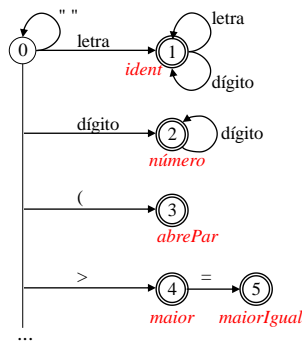
A **linguagem** reconhecida por um AFD é o conjunto de todas as seqüências de símbolos que levam o autômato do estado inicial até um dos estados finais

- Um AFD terá reconhecido uma sentença se:
  - estiver em um estado final
  - e se a entrada tiver sido consumida totalmente ou não for possível realizar uma transição com o próximo símbolo da entrada

9

## O scanner como um AFD

- O analisador léxico pode ser visto como um grande AFD



Exemplo p/ entrada: max >= 30

- $s_0 \rightarrow s_1$     sem transições " " em  $s_1$  reconhecido *Ident*
- $s_0 \rightarrow s_5$     ignora os brancos do início não para em  $s_4$  sem transições " " em  $s_5$  reconhecido *maiorIgual*
- $s_0 \rightarrow s_2$     ignora os brancos do início sem transições " " em  $s_2$  reconhecido *número*

- Depois de reconhecer um token, o scanner parte do estado  $s_0$  novamente.

10

## Implementando um AFD (ver 1)

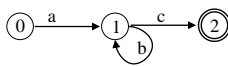
- Implementação de  $\delta$  como uma matriz

```
int[,] delta = new int[maxStates, maxSymbols];
int lastState, state = 0; // DFA starts in state 0
do {
    int sym = next symbol;
    lastState = state;
    state = delta[state, sym];
} while (state != undefined);
assert(lastState, F); // F is set of final states
return recognizedToken[lastState];
```

Este é um exemplo de algoritmo universal de reconhecimento

Exemplo:

A = a { b } c



| $\delta$ | a | b | c |
|----------|---|---|---|
| 0        | 1 | - | - |
| 1        | - | 1 | 2 |
| 2        | - | - | - |

```
int[,] delta = { {1, -1, -1},
                 {-1, 1, 2},
                 {-1, -1, -1} };
```

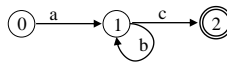
- Essa implementação pode ser muito ineficiente para um analisador real.

11

## Implementando um AFD (ver 2)

Exemplo:

A = a { b } c



- Codificação direta dos estados:

```
char ch = read();
s0: if (ch == 'a') { ch = read(); goto s1; }
    else goto err;
s1: if (ch == 'b') { ch = read(); goto s1; }
    else if (ch == 'c') { ch = read(); goto s2; }
    else goto err;
s2: return A;
err: return errorToken
```

- Que em Java poderia ser algo mais ou menos assim:

```
int state = 0;
loop:
for (;;) {
    char ch = read();
    switch (state) {
        case 0: if (ch == 'a') { state = 1; break; }
                else break loop;
        case 1: if (ch == 'b') { state = 1; break; }
                else if (ch == 'c') { state = 2; break; }
                else break loop;
        case 2: return A;
    }
}
return errorToken;
```

12

## Análise Léxica

- Esquadrinhar o código fonte, símbolo a símbolo, compondo tokens e classificando-os (segundo seu significado para a linguagem);
- Compor e gerenciar a lista de Tokens;
- Eliminar elementos desnecessários ao processo de compilação;
- Reconhecer e validar números inteiros e reais;
- Reconhecer e validar os elementos utilizados como identificadores;
- Prover um mecanismo para controle de erros amigável;
- Tratar parâmetros para compilação condicional;



Um exemplo de como ocorre processo de tokenização

## O SCANNING EM AÇÃO

## Exemplo de funcionamento

```
Prg Ex1;
{Este programa é um exemplo de
 algoritmo em LPD}
Var
  float a,b;
SubRot
int func(float x, float y)
begin
  return (x+y);
end;
begin
  write("Entre com 2 valores:");
  read(a);
  read(b);
  write(func(a,b));
end.
```

- Suponha um arquivo `fonte.lpd` contendo um programa escrito em linguagem LPD\*, como este, dado ao lado.

(\*) LPD – Linguagem de Programação Didática  
É uma linguagem estruturada, simples e de poucos elementos, com sintaxe similares ao Pascal e C

## Exemplo de funcionamento

```
PrgEx1;
{Este programa é um exemplo de
 algoritmo em LPD}
Var
  float a,b;
SubRot
int func(float x, float y)
begin
  return (x+y);
end;
begin
  write("Entre com 2 valores:");
  read(a);
  read(b);
  write(func(a,b));
end.
```

- Lembre-se que há alguns elementos que visualmente ignoramos, mas estarão presentes na leitura do arquivo, como por exemplo:

Quebras de linha: `\n`

Espaços e tabulações:

Fim de arquivo: `¢`



## Exemplo de funcionamento

```
Prg Ex1;
{Este programa é um exemplo de
 algoritmo em LPD}
Var
    float a,b;
SubRot
int func(float x, float y)
begin
    return (x+y);
end;
begin
    write("Entre com 2 valores:");
    read(a);
    read(b);
    write(func(a,b));
end.
```

- Para realizar esse processamento, precisamos controlar:
  - i. Caractere lido;
  - ii. O token que estamos formando;
  - iii. Número da linha; e
  - iv. Lista de tokens.

## Exemplo de funcionamento

```
PrgEx1;
{Este programa é um exemplo de
 algoritmo em LPD}
Var
    float a,b;
SubRot
int func(float x, float y)
begin
    return (x+y);
end;
begin
    write("Entre com 2 valores:");
    read(a);
    read(b);
    write(func(a,b));
end.
```

Caracter:

Token:

Classif:

#Linha:

## Exemplo de funcionamento

```

PrgEx1;
{Este programa é um exemplo de
 algoritmo em LPD}
Var
  a,b:float;
SubRot
  int func(float x, float y)
  begin
    return x+y;
  end;
begin
  write("Entre com 2 valores:");
  read(a);
  read(b);
  write(func(a,b));
end.

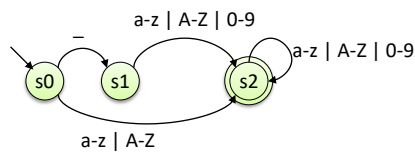
```

Caracter: P

Token:

Classif:

#Linha: 1



## Exemplo de funcionamento

```

PrgEx1;
{Este programa é um exemplo de
 algoritmo em LPD}
Var
  a,b:float;
SubRot
  int func(float x, float y)
  begin
    return x+y;
  end;
begin
  write("Entre com 2 valores:");
  read(a);
  read(b);
  write(func(a,b));
end.

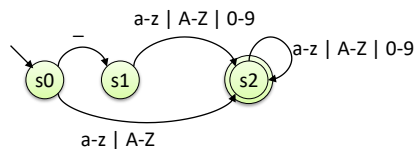
```

Caracter: P

Token: P

Classif:

#Linha: 1



# Exemplo de funcionamento

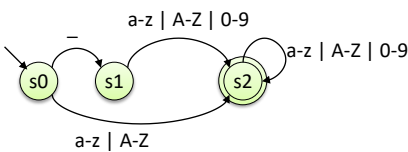
```
PrgEx1;
{Este programa é um exemplo de
 algoritmo em LPD}
Var
  a,b:float;
SubRot
  func(float x, float y)
  begin
    return(x+y);
  end;
begin
  write("Entre com 2 valores:");
  read(a);
  read(b);
  write(func(a,b));
end.
```

Caracter: r

Token: Pr

Classif:

#Linha: 1



# Exemplo de funcionamento

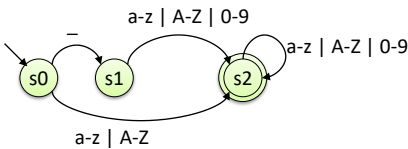
```
PrgEx1;
{Este programa é um exemplo de
 algoritmo em LPD}
Var
  a,b:float;
SubRot
  func(float x, float y)
  begin
    return(x+y);
  end;
begin
  write("Entre com 2 valores:");
  read(a);
  read(b);
  write(func(a,b));
end.
```

Caracter: g

Token: Prg

Classif:

#Linha: 1



# Exemplo de funcionamento

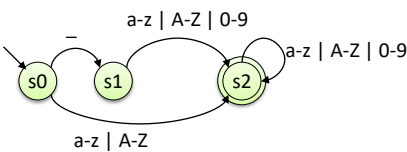
```
Proc Ex1;
{Este programa é um exemplo de
 algoritmo em LPD}
Var
  a,b: float;
SubRot
int func(float x, float y)
begin
  return (x+y);
end;
begin
  write("Entre com 2 valores:");
  read(a);
  read(b);
  write(func(a,b));
end.
```

Caracter: ¤

Token: Prg

Classif:

#Linha: 1



# Exemplo de funcionamento

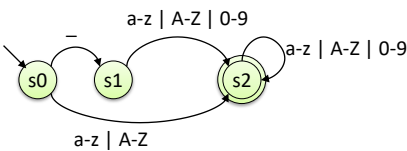
```
Proc Ex1;
{Este programa é um exemplo de
 algoritmo em LPD}
Var
  a,b: float;
SubRot
int func(float x, float y)
begin
  return (x+y);
end;
begin
  write("Entre com 2 valores:");
  read(a);
  read(b);
  write(func(a,b));
end.
```

Caracter: ¤

Token: Prg

Classif: sPRG

#Linha: 1



# Exemplo de funcionamento

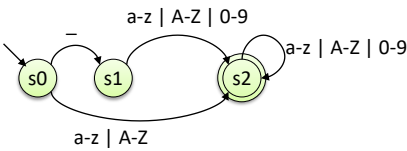
```
Proc Ex1;
{Este programa é um exemplo de
 algoritmo em LPD}
Var
  a,b:float;
SubRot
int func(float x, float y)
begin
  return(x+y);
end;
begin
  write("Entre com 2 valores:");
  read(a);
  read(b);
  write(func(a,b));
end.
```

Caracter: ¤

Token:

Classif:

#Linha: 1



| Token | Classif | Lin # |
|-------|---------|-------|
| Prg   | sPRG    | 1     |

# Exemplo de funcionamento

```
Proc Ex1;
{Este programa é um exemplo de
 algoritmo em LPD}
Var
  a,b:float;
SubRot
int func(float x, float y)
begin
  return(x+y);
end;
begin
  write("Entre com 2 valores:");
  read(a);
  read(b);
  write(func(a,b));
end.
```

Caracter: ¤

Token:

Classif:

#Linha: 1

## Exemplo de funcionamento

```

Prg: Ex1;
{Este programa é um exemplo de
 algoritmo em LPD}
Var
  a,b: float;
SubRot
  int func(float x, float y)
  begin
    return (x+y);
  end;
begin
  write("Entre com 2 valores:");
  read(a);
  read(b);
  write(func(a,b));
end.

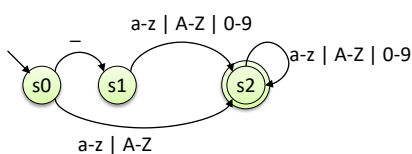
```

Caracter: E

Token:

Classif:

#Linha: 1



## Exemplo de funcionamento

```

Prg: Ex1;
{Este programa é um exemplo de
 algoritmo em LPD}
Var
  a,b: float;
SubRot
  int func(float x, float y)
  begin
    return (x+y);
  end;
begin
  write("Entre com 2 valores:");
  read(a);
  read(b);
  write(func(a,b));
end.

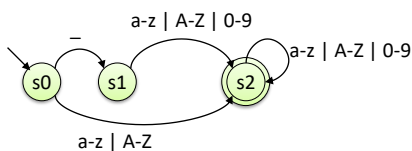
```

Caracter: E

Token: E

Classif:

#Linha: 1



# Exemplo de funcionamento

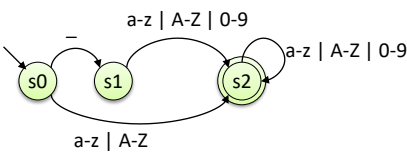
```
PrgEx1;
{Este programa é um exemplo de
 algoritmo em LPD}
Var
  a,b:float;
SubRot
  func(float x, float y)
  begin
    return(x+y);
  end;
begin
  write("Entre com 2 valores:");
  read(a);
  read(b);
  write(func(a,b));
end.
```

Caracter: x

Token: Ex

Classif:

#Linha: 1



# Exemplo de funcionamento

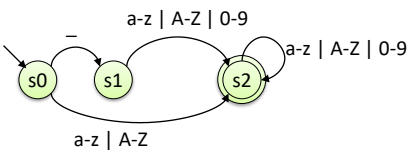
```
PrgEx1;
{Este programa é um exemplo de
 algoritmo em LPD}
Var
  a,b:float;
SubRot
  func(float x, float y)
  begin
    return(x+y);
  end;
begin
  write("Entre com 2 valores:");
  read(a);
  read(b);
  write(func(a,b));
end.
```

Caracter: 1

Token: Ex1

Classif:

#Linha: 1



## Exemplo de funcionamento

```

Prg#Ex1;
{Este#programa#é#um#exemplo#de#
#algoritmo#em#LPD}
Var
  ##float#a,b;
SubRot
  int#func(float#x,#float#y)
  begin
    ##return#(x+y);
  end;
begin
  ##write("Entre com 2 valores:");
  ##read(a);
  ##read(b);
  ##write(func(a,b));
end.##

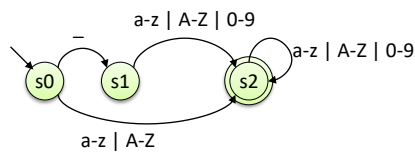
```

Caracter: ;

Token: Ex1

Classif:

#Linha: 1



## Exemplo de funcionamento

```

Prg#Ex1;
{Este#programa#é#um#exemplo#de#
#algoritmo#em#LPD}
Var
  ##float#a,b;
SubRot
  int#func(float#x,#float#y)
  begin
    ##return#(x+y);
  end;
begin
  ##write("Entre com 2 valores:");
  ##read(a);
  ##read(b);
  ##write(func(a,b));
end.##

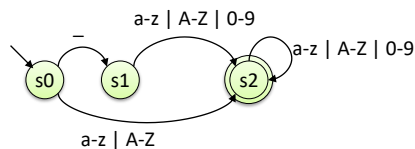
```

Caracter: ;

Token: Ex1

Classif: sID

#Linha: 1





# Exemplo de funcionamento

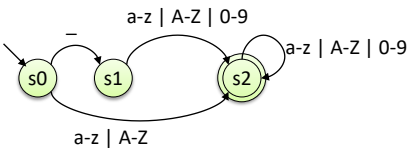
```
PrgEx1;
{Este programa é um exemplo de
 algoritmo em LPD}
Var
  a,b;
SubRot
  int func(float x, float y)
  begin
    return (x+y);
  end;
begin
  write("Entre com 2 valores:");
  read(a);
  read(b);
  write(func(a,b));
end.
```

Caracter: ;

Token:

Classif:

#Linha: 1



| Token | Classif | Lin # |
|-------|---------|-------|
| Ex1   | sID     | 1     |

# Exemplo de funcionamento

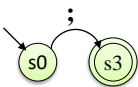
```
PrgEx1;
{Este programa é um exemplo de
 algoritmo em LPD}
Var
  a,b;
SubRot
  int func(float x, float y)
  begin
    return (x+y);
  end;
begin
  write("Entre com 2 valores:");
  read(a);
  read(b);
  write(func(a,b));
end.
```

Caracter: ;

Token:

Classif:

#Linha: 1



## Exemplo de funcionamento

```

PrgEx1;
{Este programa é um exemplo de
 algoritmo em LPD}
Var
  a,b;
SubRot
  int func(float x, float y)
  begin
    return (x+y);
  end;
begin
  write("Entre com 2 valores:");
  read(a);
  read(b);
  write(func(a,b));
end.

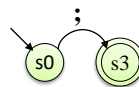
```

Caracter: ;

Token: ;

Classif:

#Linha: 1



## Exemplo de funcionamento

```

PrgEx1;
{Este programa é um exemplo de
 algoritmo em LPD}
Var
  a,b;
SubRot
  int func(float x, float y)
  begin
    return (x+y);
  end;
begin
  write("Entre com 2 valores:");
  read(a);
  read(b);
  write(func(a,b));
end.

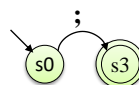
```

Caracter: ;

Token: ;

Classif: sPtoVirg

#Linha: 1



# Exemplo de funcionamento

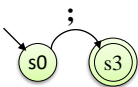
```
PrgEx1;
{Este programa é um exemplo de
 algoritmo em LPD}
Var
  a,b;
SubRot
int func(float x, float y)
begin
  return (x+y);
end;
begin
  write("Entre com 2 valores:");
  read(a);
  read(b);
  write(func(a,b));
end.
ç
```

Caracter: ;

Token:

Classif:

#Linha: 1



| Token | Classif  | Lin # |
|-------|----------|-------|
| ;     | sPtoVirg | 1     |

# Exemplo de funcionamento

```
PrgEx1;
{Este programa é um exemplo de
 algoritmo em LPD}
Var
  a,b;
SubRot
int func(float x, float y)
begin
  return (x+y);
end;
begin
  write("Entre com 2 valores:");
  read(a);
  read(b);
  write(func(a,b));
end.
ç
```

Caracter: ¶

Token:

Classif:

#Linha: 1

## Exemplo de funcionamento

```

PrgEx1;
{Este programa é um exemplo de
 algoritmo em LPD}
Var
  a,b:float;
SubRot
int func(float x, float y)
begin
  return (x+y);
end;
begin
  write("Entre com 2 valores:");
  read(a);
  read(b);
  write(func(a,b));
end.

```

Caracter: {

Token:

Classif:

#Linha: 2

## Exemplo de funcionamento

```

PrgEx1;
{Este programa é um exemplo de
 algoritmo em LPD}
Var
  a,b:float;
SubRot
int func(float x, float y)
begin
  return (x+y);
end;
begin
  write("Entre com 2 valores:");
  read(a);
  read(b);
  write(func(a,b));
end.

```

Caracter: E

Token:

Classif:

#Linha: 2

## Exemplo de funcionamento

```

Prg#Ex1;#
{Este#programa#é#um#exemplo#de#
#algoritmo#em#LPD}#
Var#
###float#a,b;#
SubRot#
int#func(float#x,#float#y)#
begin#
###return#(x+y);#
end;#
begin#
###write("Entre com 2 valores:");#
###read(a);#
###read(b);#
###write(func(a,b));#
end.##

```

Caracter: s

Token:

Classif:

#Linha: 2

## Exemplo de funcionamento

```

Prg#Ex1;#
{Este#programa#é#um#exemplo#de#
#algoritmo#em#LPD}#
Var#
###float#a,b;#
SubRot#
int#func(float#x,#float#y)#
begin#
###return#(x+y);#
end;#
begin#
###write("Entre com 2 valores:");#
###read(a);#
###read(b);#
###write(func(a,b));#
end.##

```

Caracter: P

Token:

Classif:

#Linha: 1

## Exemplo de funcionamento

```

PrgEx1;
{Este programa é um exemplo de
 algoritmo em LPD}
Var
  a,b:float;
SubRot
intfunc(floatx, floaty)
begin
  return(x+y);
end;
begin
  write("Entre com 2 valores:");
  read(a);
  read(b);
  write(func(a,b));
end.

```

Caracter: e

Token:

Classif:

#Linha: 2

## Exemplo de funcionamento

```

PrgEx1;
{Este programa é um exemplo de
 algoritmo em LPD}
Var
  a,b:float;
SubRot
intfunc(floatx, floaty)
begin
  return(x+y);
end;
begin
  write("Entre com 2 valores:");
  read(a);
  read(b);
  write(func(a,b));
end.

```

Caracter: x

Token:

Classif:

#Linha: 2

## Exemplo de funcionamento

```
PrgEx1;
{Este programa é um exemplo de
algoritmo em LPD}
Var
  a,b:float;
SubRot
  int func(float x, float y)
  begin
    return (x+y);
  end;
begin
  write("Entre com 2 valores:");
  read(a);
  read(b);
  write(func(a,b));
end.
```

Caracter: p

Token:

Classif:

#Linha: 2

## Exemplo de funcionamento

```
PrgEx1;
{Este programa é um exemplo de
algoritmo em LPD}
Var
  a,b:float;
SubRot
  int func(float x, float y)
  begin
    return (x+y);
  end;
begin
  write("Entre com 2 valores:");
  read(a);
  read(b);
  write(func(a,b));
end.
```

Caracter: r

Token:

Classif:

#Linha: 2

## Exemplo de funcionamento

```
PrgEx1;
{Este programa é um exemplo de
algoritmo em LFD}
Var
  a,b:float;
SubRot
  int func(float x, float y)
  begin
    return (x+y);
  end;
begin
  write("Entre com 2 valores:");
  read(a);
  read(b);
  write(func(a,b));
end.
```

Caracter: o

Token:

Classif:

#Linha: 2

## Exemplo de funcionamento

```
PrgEx1;
{Este programa é um exemplo de
algoritmo em LFD}
Var
  a,b:float;
SubRot
  int func(float x, float y)
  begin
    return (x+y);
  end;
begin
  write("Entre com 2 valores:");
  read(a);
  read(b);
  write(func(a,b));
end.
```

Caracter: g

Token:

Classif:

#Linha: 2



# Exemplo de funcionamento

```
PrgEx1;
{Este programa é um exemplo de
algoritmo em LPD}
Var
  a,b:float;
SubRot
  int func(float x, float y)
  begin
    return (x+y);
  end;
begin
  write("Entre com 2 valores:");
  read(a);
  read(b);
  write(func(a,b));
end.
```

Caracter: r

Token:

Classif:

#Linha: 2

# Exemplo de funcionamento

```
PrgEx1;
{Este programa é um exemplo de
algoritmo em LPD}
Var
  a,b:float;
SubRot
  int func(float x, float y)
  begin
    return (x+y);
  end;
begin
  write("Entre com 2 valores:");
  read(a);
  read(b);
  write(func(a,b));
end.
```

Caracter: a

Token:

Classif:

#Linha: 2

# Exemplo de funcionamento

```
PrgEx1;
{Este programa é um exemplo de
 algoritmo em LPD}
Var
  a,b:float;
SubRot
intfunc(floatx, floaty)
begin
  return(x+y);
end;
begin
  write("Entre com 2 valores:");
  read(a);
  read(b);
  write(func(a,b));
end.
```

Caracter: m

Token:

Classif:

#Linha: 2

# Exemplo de funcionamento

```
PrgEx1;
{Este programa é um exemplo de
 algoritmo em LPD}
Var
  a,b:float;
SubRot
intfunc(floatx, floaty)
begin
  return(x+y);
end;
begin
  write("Entre com 2 valores:");
  read(a);
  read(b);
  write(func(a,b));
end.
```

Caracter: a

Token:

Classif:

#Linha: 2

## Exemplo de funcionamento

```

PrgEx1;
{Este programa é um exemplo de
 algoritmo em LPD}
Var
  a,b:float;
SubRot
intfunc(floatx, floaty)
begin
  return(x+y);
end;
begin
  write("Entre com 2 valores:");
  read(a);
  read(b);
  write(func(a,b));
end.

```

Caracter: 8

Token:

Classif:

#Linha: 2

## Exemplo de funcionamento

```

PrgEx1;
{Este programa é um exemplo de
 algoritmo em LPD}
Var
  a,b:float;
SubRot
intfunc(floatx, floaty)
begin
  return(x+y);
end;
begin
  write("Entre com 2 valores:");
  read(a);
  read(b);
  write(func(a,b));
end.

```

Caracter: 8

Token:

Classif:

#Linha: 2

## Exemplo de funcionamento

```
Prg#Ex1;¶
{Este#programa#é#um#exemplo#de#¶
#algoritmo#em#LPD}¶
Var¶
###float#a,b;¶
SubRot¶
int#func(float#x,#float#y)¶
begin¶
###return#(x+y);¶
end;¶
begin¶
###write("Entre com 2 valores:");¶
###read(a);¶
###read(b);¶
###write(func(a,b));¶
end.##¢
```

Caracter: d

Token:

Classif:

#Linha: 2

## Exemplo de funcionamento

```
Prg#Ex1;¶
{Este#programa#é#um#exemplo#de#¶
#algoritmo#em#LPD}¶
Var¶
###float#a,b;¶
SubRot¶
int#func(float#x,#float#y)¶
begin¶
###return#(x+y);¶
end;¶
begin¶
###write("Entre com 2 valores:");¶
###read(a);¶
###read(b);¶
###write(func(a,b));¶
end.##¢
```

Caracter: e

Token:

Classif:

#Linha: 2

## Exemplo de funcionamento

```
PrgEx1;
{Este programa é um exemplo de
algoritmo em LPD}
Var
  a,b:float;
SubRot
  int func(float x, float y)
  begin
    return (x+y);
  end;
begin
  write("Entre com 2 valores:");
  read(a);
  read(b);
  write(func(a,b));
end.
```

Caracter: ¶

Token:

Classif:

#Linha: 2

## Exemplo de funcionamento

```
PrgEx1;
{Este programa é um exemplo de
algoritmo em LPD}
Var
  a,b:float;
SubRot
  int func(float x, float y)
  begin
    return (x+y);
  end;
begin
  write("Entre com 2 valores:");
  read(a);
  read(b);
  write(func(a,b));
end.
```

Caracter: ¶

Token:

Classif:

#Linha: 3

# Exemplo de funcionamento

```
PrgEx1;
{Este programa é um exemplo de
algoritmo em LPD}
Var
  a,b:float;
SubRot
  int func(float x, float y)
  begin
    return (x+y);
  end;
begin
  write("Entre com 2 valores:");
  read(a);
  read(b);
  write(func(a,b));
end.
```

Caracter: ¤

Token:

Classif:

#Linha: 3

# Exemplo de funcionamento

```
PrgEx1;
{Este programa é um exemplo de
algoritmo em LPD}
Var
  a,b:float;
SubRot
  int func(float x, float y)
  begin
    return (x+y);
  end;
begin
  write("Entre com 2 valores:");
  read(a);
  read(b);
  write(func(a,b));
end.
```

Caracter: a

Token:

Classif:

#Linha: 3

## Exemplo de funcionamento

```

PrgEx1;
{Este programa é um exemplo de
algoritmo em LPD}
Var
  a,b:float;
SubRot
  int func(float x, float y)
  begin
    return (x+y);
  end;
begin
  write("Entre com 2 valores:");
  read(a);
  read(b);
  write(func(a,b));
end.

```

Caracter: l

Token:

Classif:

#Linha: 3

## Exemplo de funcionamento

```

PrgEx1;
{Este programa é um exemplo de
algoritmo em LPD}
Var
  a,b:float;
SubRot
  int func(float x, float y)
  begin
    return (x+y);
  end;
begin
  write("Entre com 2 valores:");
  read(a);
  read(b);
  write(func(a,b));
end.

```

Caracter: g

Token:

Classif:

#Linha: 3

# Exemplo de funcionamento

```
Prg#Ex1;¶
{Este#programa#é#um#exemplo#de#
#algoritmo#em#L#D}¶
Var¶
###float#a,b;¶
SubRot¶
int#func(float#x,#float#y)¶
begin¶
###return#(x+y);¶
end;¶
begin¶
###write("Entre com 2 valores:");¶
###read(a);¶
###read(b);¶
###write(func(a,b));¶
end.##¢
```

Caracter: P

Token:

Classif:

#Linha: 3

# Exemplo de funcionamento

```
Prg#Ex1;¶
{Este#programa#é#um#exemplo#de#
#algoritmo#em#L#D}¶
Var¶
###float#a,b;¶
SubRot¶
int#func(float#x,#float#y)¶
begin¶
###return#(x+y);¶
end;¶
begin¶
###write("Entre com 2 valores:");¶
###read(a);¶
###read(b);¶
###write(func(a,b));¶
end.##¢
```

Caracter: D

Token:

Classif:

#Linha: 3



# Exemplo de funcionamento

```
PrgEx1;
{Este programa é um exemplo de
 algoritmo em LPD}
Var
  a,b:float;
SubRot
intfunc(floatx, floaty)
begin
  return(x+y);
end;
begin
  write("Entre com 2 valores:");
  read(a);
  read(b);
  write(func(a,b));
end.
```

Caracter: }

Token:

Classif:

#Linha: 3

# Exemplo de funcionamento

```
PrgEx1;
{Este programa é um exemplo de
 algoritmo em LPD}
Var
  a,b:float;
SubRot
intfunc(floatx, floaty)
begin
  return(x+y);
end;
begin
  write("Entre com 2 valores:");
  read(a);
  read(b);
  write(func(a,b));
end.
```

Caracter: }

Token:

Classif:

#Linha: 3

## Exemplo de funcionamento

```

PrgEx1;
{Este programa é um exemplo de
 algoritmo em LPD}
Var
  a,b;
SubRot
  func(float x, float y)
  begin
    return (x+y);
  end;
begin
  write("Entre com 2 valores:");
  read(a);
  read(b);
  write(func(a,b));
end.

```

Caracter: ¶

Token:

Classif:

#Linha: 4

## Exemplo de funcionamento

```

PrgEx1;
{Este programa é um exemplo de
 algoritmo em LPD}
Var
  a,b;
SubRot
  func(float x, float y)
  begin
    return (x+y);
  end;
begin
  write("Entre com 2 valores:");
  read(a);
  read(b);
  write(func(a,b));
end.

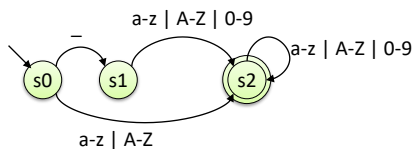
```

Caracter: V

Token:

Classif:

#Linha: 4



# Exemplo de funcionamento

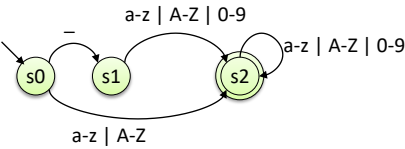
```
PrgEx1;
{Este programa é um exemplo de
 algoritmo em LPD}
Var
  float a,b;
SubRot
  func(float x, float y)
  begin
    return(x+y);
  end;
begin
  write("Entre com 2 valores:");
  read(a);
  read(b);
  write(func(a,b));
end.
```

Caracter: V

Token: V

Classif:

#Linha: 4



# Exemplo de funcionamento

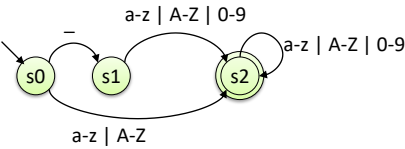
```
PrgEx1;
{Este programa é um exemplo de
 algoritmo em LPD}
Var
  float a,b;
SubRot
  func(float x, float y)
  begin
    return(x+y);
  end;
begin
  write("Entre com 2 valores:");
  read(a);
  read(b);
  write(func(a,b));
end.
```

Caracter: a

Token: Va

Classif:

#Linha: 4



# Exemplo de funcionamento

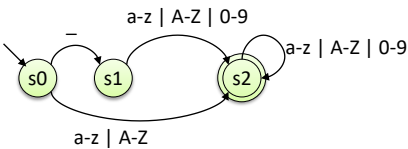
```
Prg#Ex1;{Este#programa#é#um#exemplo#de#algoritmo#em#LPD}Var##float#A,B;SubRot#int#func(float#x,float#y){begin##return#(x+y);end;}begin##write("Entre com 2 valores:");##read(A);##read(B);##write(func(A,B));end.##
```

Caracter: r

Token: Var

Classif:

#Linha: 4



# Exemplo de funcionamento

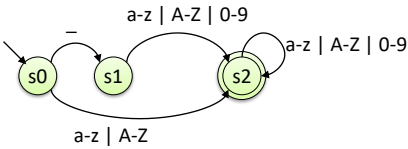
```
Prg#Ex1;{Este#programa#é#um#exemplo#de#algoritmo#em#LPD}Var##float#A,B;SubRot#int#func(float#x,float#y){begin##return#(x+y);end;}begin##write("Entre com 2 valores:");##read(A);##read(B);##write(func(A,B));end.##
```

Caracter: ¶

Token: Var

Classif:

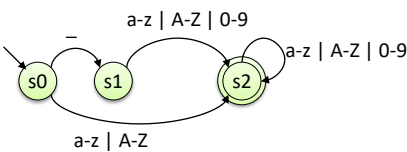
#Linha: 4



# Exemplo de funcionamento

```
PrgEx1;
{Este programa é um exemplo de
 algoritmo em LPD}
Var
  float a,b;
SubRot
int func(float x, float y)
begin
  return(x+y);
end;
begin
  write("Entre com 2 valores:");
  read(a);
  read(b);
  write(func(a,b));
end.
```

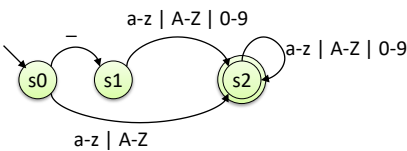
Caracter: `{`  
Token: Var  
Classif: sVAR  
#Linha: 4



# Exemplo de funcionamento

```
PrgEx1;
{Este programa é um exemplo de
 algoritmo em LPD}
Var
  float a,b;
SubRot
int func(float x, float y)
begin
  return(x+y);
end;
begin
  write("Entre com 2 valores:");
  read(a);
  read(b);
  write(func(a,b));
end.
```

Caracter: `{`  
Token:  
Classif:  
#Linha: 4



| Token | Classif | Lin # |
|-------|---------|-------|
| Var   | sVAR    | 4     |

# Exemplo de funcionamento

```
Prg#Ex1;¶
{Este#programa#é#um#exemplo#de#
#algoritmo#em#LPD}¶
Var¶
#float#a,b;¶
SubRot¶
int#func(float#x,#float#y)¶
begin¶
#return#(x+y);¶
end;¶
begin¶
#write("Entre com 2 valores:");¶
#read(a);¶
#read(b);¶
#write(func(a,b));¶
end.#¶
```

Caracter: ¶

Token:

Classif:

#Linha: 5

# Exemplo de funcionamento

```
Prg#Ex1;¶
{Este#programa#é#um#exemplo#de#
#algoritmo#em#LPD}¶
Var¶
#float#a,b;¶
SubRot¶
int#func(float#x,#float#y)¶
begin¶
#return#(x+y);¶
end;¶
begin¶
#write("Entre com 2 valores:");¶
#read(a);¶
#read(b);¶
#write(func(a,b));¶
end.#¶
```

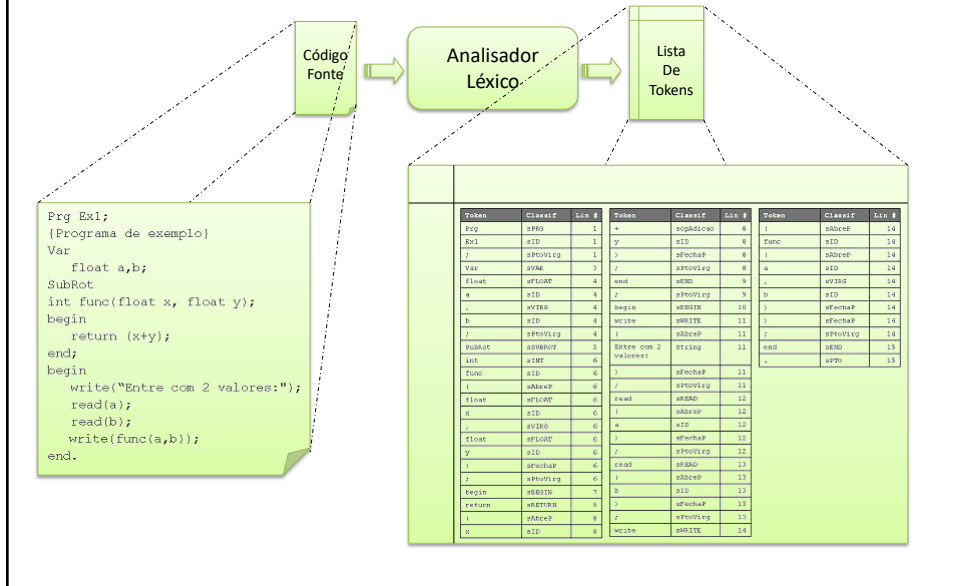
Caracter: ¶

Token:

Classif:

#Linha: 16

## 1ª etapa, concluída!



## Autômato de tokenização

- Reconhecimento de palavras reservadas e identificadores
- Tokens formados por um único caractere
- Números inteiros e números reais
- Op. Relacionais e de Atribuição

