Compiladores

Análise Léxica

Cristiano Lehrer, M.Sc.

Copyright © 2009/2024 • ybadoo.com.br

Introdução (1/3)

- Análise léxica é a primeira fase do compilador.
- A função do analisador léxico, também denominado ***scanner***, é:
  - Fazer a leitura do programa fonte, caractere a caractere, e traduzi-lo para uma sequência de símbolos léxicos, também chamados *tokens*.
- Exemplos de símbolos léxicos:
  - Palavras reservadas.
  - Identificadores.
  - Constantes.
  - Operadores da linguagem.

Copyright © 2009/2024 • ybadoo.com.br

ybadoo

Introdução (2/3)

- Durante o processo de análise léxica, são desprezados caracteres não significativos como espaços em branco e comentários.
- Além de reconhecer os símbolos léxicos, o analisador também realiza outras funções:
  - Armazena alguns desses símbolos (identificadores e constantes) em tabelas internas.
  - Indica a ocorrência de erros léxicos.
- A sequência de *tokens* produzida pelo analisador léxico é utilizada como entrada pelo módulo seguinte do compilador, o analisador sintático.

Copyright © 2009/2024 • ybadoo.com.br

ybadoo

Introdução (3/3)

- O programa fonte é visto de forma diferente pelo analisadores:
  - Analisador léxico:
    - O programa fonte é uma sequência de palavras de uma **linguagem regular**.
  - Analisador sintático:
    - Essa sequência de *tokens* constitui uma sentença de um **linguagem livre do contexto**.

Copyright © 2009/2024 • ybadoo.com.br

ybadoo

Tokens (1/3)

- *Tokens* ou símbolos léxicos são as unidades básicas do texto do programa.
- Cada *token* é representado internamente por três informações:
  - Classe do *token*:
    - Representa o tipo do *token* reconhecido, como identificador, operador.
  - Valor do *token*:
    - Depende da classe. Por exemplo, para constantes numéricas, o valor do *token* pode ser o número inteiro representado pela constante.
  - Posição do *token*:
    - Indica o local do texto fonte (linha e coluna) onde ocorreu o *token*.
    - Informação utilizada para indicar o local de erros.

Copyright © 2009/2024 • ybadoo.com.br

ybadoo

Tokens (2/3)

- Em função do campo valor do *token*, os *tokens* podem ser divididos em dois grupos:
  - *Tokens* simples:
    - Não têm um valor associado porque a classe do *token* descreve-o completamente.
    - Correspondem a elementos fixos da linguagem.
  - *Tokens* com argumento:
    - Têm um valor associado.
    - Correspondem aos elementos da linguagem definidos pelo programador, como por exemplo, identificadores, constantes numéricas e cadeias de caracteres.

Copyright © 2009/2024 • ybadoo.com.br

ybadoo

Tokens (3/3)

- Exemplo:
  - `while I < 100 do I := J + I;`
    - `[while, ][id, 7][<, ][cte, 13][do, ][id, 7][:=, ][id, 12][+, ][id, 7][;, ]`
    - Para simplificar, os tokens estão representados por pares (omitiu-se a posição).
    - Identificadores e constantes numéricas estão representados pelo par [classe do token, índice na tabela).
    - As classes para palavras reservadas constituem-se em abreviações dessas, não sendo necessário passar seus valores para o analisador sintático.
    - Para delimitadores e operadores, a classe é o próprio valor do token.
    - Usualmente, os compiladores representam a classe de um token por um número inteiro para tornar a representação mais compacta.

Copyright © 2009/2024 • ybadoo.com.br

ybadoo

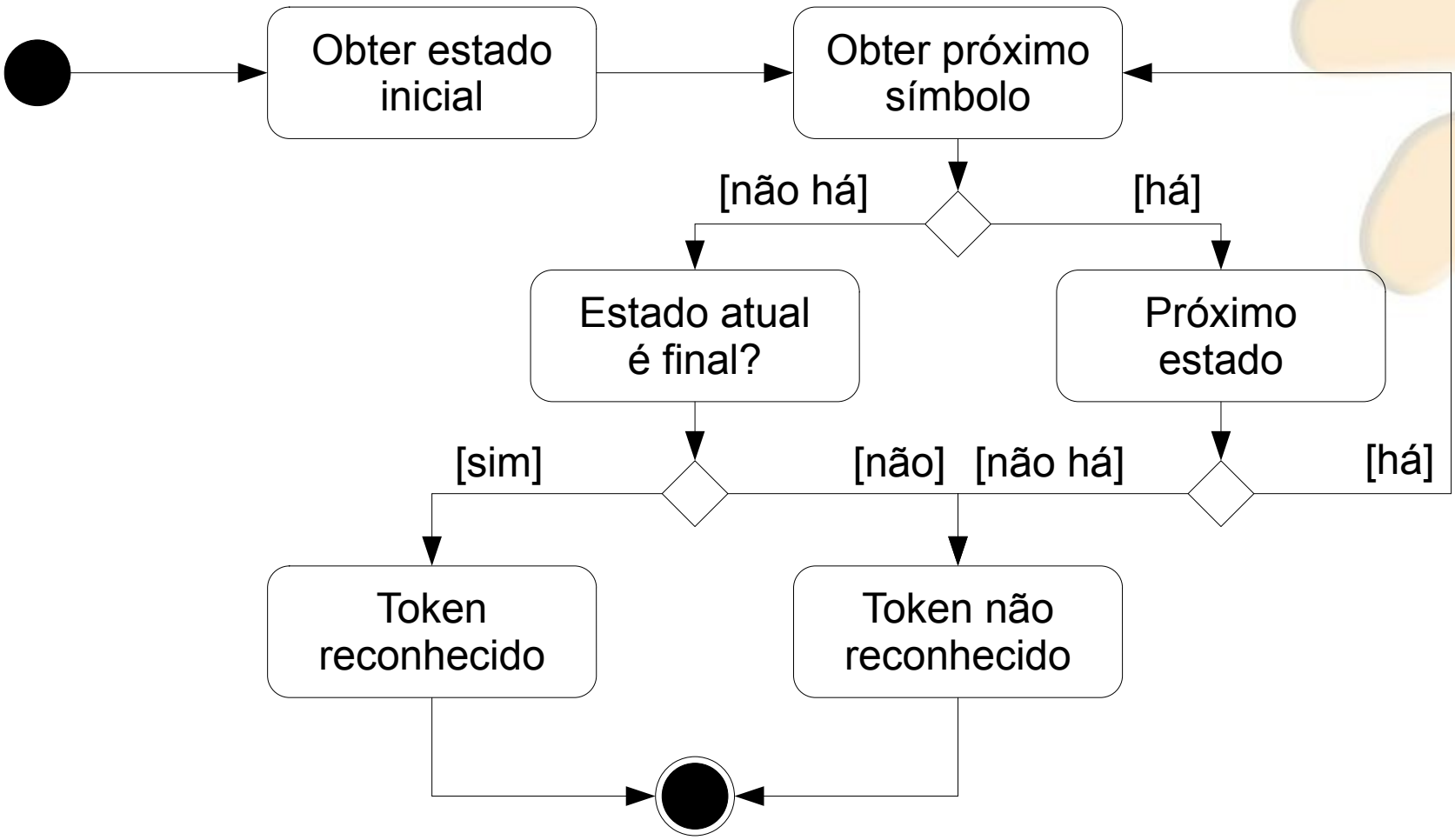
Tabela de Símbolos

- Estrutura de dados gerado pelo compilador com o objetivo de armazenar informações sobre os nomes (identificadores de variáveis, de parâmetros, de funções, de procedimentos, entre outras) definidos no programa fonte.
- Associa atributos (tipo, escopo, limites no caso de vetores e número de parâmetros no caso de funções) aos nomes definidos pelo programador.
- Começa a ser construída durante a análise léxica, quando os identificadores são reconhecidos.
- Toda vez que um identificador é reconhecido no programa fonte, a Tabela de Símbolos é consultada, a fim de verificar se o nome já está registrado; caso não esteja, é feita sua inserção na tabela.

Copyright © 2009/2024 • ybadoo.com.br

ybadoo

Analisadores Léxicos (1/2)



Copyright © 2009/2024 • ybadoo.com.br

Analisadores Léxicos (2/2)

- Algoritmo para um analisador léxico:

```
RECONHECE(M, T)
1 s ← ESTADO-INICIAL(M)
2 while TEM-SÍMBOLOS(T)
3 do c ← PRÓXIMO-SÍMBOLO(T)
4   if EXISTE-PRÓXIMO-ESTADO(M, s, c)
5     then s ← PRÓXIMO-ESTADO(M, s, c)
6     else return false
7 if ESTADO-FINAL(M, s)
8   then return true
9   then return false
```


Copyright © 2009/2024 • ybadoo.com.br

ybadoo

Linguagem SIMPLE (1/2)

- Cada instrução SIMPLE consiste em um número de linha e um comando.
- Os números de linha devem aparecer em ordem crescente.
- Devem ser usadas apenas letras minúsculas.
- O nome de variável tem uma única letra, sendo do tipo inteiro.
- Operadores aritméticos:
  - Adição (+), subtração (-), multiplicação (*), divisão (/) e resto da divisão (%).
- Operadores relacionais:
  - Maior (>), maior ou igual (>=), menor (<), menor ou igual (<=), igual (==), desigual (!=).

Copyright © 2009/2024 • ybadoo.com.br

ybadoo

Linguagem SIMPLE (2/2)

Comando	Instrução de Exemplo	Descrição
rem	50 rem isto é um comentário	Qualquer texto seguindo o comando **rem** é apenas para propósitos de documentação e é ignorado pelo compilador.
input	30 input x	Exibe um ponto de interrogação para pedir ao usuário para inserir um inteiro. Lê esse inteiro do teclado e armazena o inteiro em **x**.
let	80 let u = j – 36	Atribui a **u** o valor de **j – 36**. Observe que somente serão aceitas expressões simples do lado direito do sinal de atribuição.
print	10 print w	Exibe o valor de **w**.
goto	70 goto 45	Transfere o comando do programa para a linha 45.
if	35 if i == x goto 80	Compara **i** e **x** para igualdade e transfere o controle do programa para a linha **80** se a condição for verdadeira; caso contrário, continua a execução com a próxima instrução.
end	99 end	Termina a execução do programa.

Copyright © 2009/2024 • ybadoo.com.br

ybadoo

Análise Léxica da Linguagem SIMPLE (1/3)

- Delimitadores:
  - Nova linha (LF) → 10
  - Fim de texto (ETX) → 03
- Operadores:
  - Atribuição (=) → 11
- Operadores aritméticos:
  - Adição (+) → 21
  - Subtração (-) → 22
  - Multiplicação (*) → 23
  - Divisão (/) → 24
  - Resto da divisão (%) → 25

Copyright © 2009/2024 • ybadoo.com.br

ybadoo

Análise Léxica da Linguagem SIMPLE (2/3)

- Operadores relacionais:
  - Igual (==) → 31
  - Desigual (!=) → 32
  - Maior (>) → 33
  - Menor (<) → 34
  - Maior ou igual (>=) → 35
  - Menor ou igual (<=) → 36
- Identificadores:
  - Variáveis → 41
- Constantes:
  - Constantes numéricas inteiras → 51

Copyright © 2009/2024 • ybadoo.com.br

ybadoo

Análise Léxica da Linguagem SIMPLE (3/3)

- Palavras reservadas:
  - rem → 61
  - input → 62
  - let → 63
  - print → 64
  - goto → 65
  - if → 66
  - end → 67

Copyright © 2009/2024 • ybadoo.com.br

ybadoo

Exemplo (1/2)

- Programa na linguagem SIMPLE:

```
1000 rem Encontrar o maior de dois inteiros <ENTER>
1100 input x <ENTER>
1200 input y <ENTER>
1300 if x < y goto 2000 <ENTER>
1400 print x <ENTER>
1500 goto 3000 <ENTER>
2000 print y <ENTER>
3000 end <ETX>
```


Copyright © 2009/2024 • ybadoo.com.br

ybadoo

Exemplo (2/2)

- Resultado do analisador léxico:

```
[51, 0, (1, 1)][61, , (1, 6)][10, , (1, 44)]

[51, 1, (2, 1)][62, , (2, 6)][41, 2, (2, 12)][10, , (2, 13)]

[51, 3, (3, 1)][62, , (3, 6)][41, 4, (3, 12)][10, , (3, 13)]

[51, 5, (4, 1)][66, , (4, 6)][41, 2, (4, 9)][34, , (4, 11)]
[41, 4, (4, 13)][65, , (4, 15)][51, 6, (4, 20)][10, , (4, 24)]

[51, 7, (5, 1)][64, , (5, 6)][41, 2, (5, 12)][10, , (5, 13)]

[51, 8, (6, 1)][65, , (6, 6)][51, 9, (6, 11)][10, , (6, 15)]

[51, 6, (7, 1)][64, , (7, 6)][41, 4, (7, 12)][10, , (7, 13)]

[51, 9, (8, 1)][67, , (8, 6)][3, , (8, 9)]
```

```
0 : 1000
1 : 1100
2 : x
3 : 1200
4 : y
5 : 1300
6 : 2000
7 : 1400
8 : 1500
9 : 3000
```


Copyright © 2009/2024 • ybadoo.com.br

ybadoo