UNIVERSIDADE ESTADUAL PAULISTA
“JÚLIO DE MESQUITA FILHO”

Analisador Léxico

Beatriz Ferreira de Lima
Douglas Brandão dos Santos
Instituto de Biociências, Letras e Ciências Exatas — IBILCE, Universidade Estadual Paulista "Júlio de Mesquita Filho" — UNESP, Rua Cristóvão Colombo 2265, Jd.Nazareth, 15054-000, São José do Rio Preto – SP, Brasil. E-mail: beatriz.f.lima@unesp.br, douglas.brandao@unesp.br

20 de Setembro de 2019

Resumo

Este documento destina-se a descrever a linguagem desenvolvida pelos autores, bem como explicar as particularidades e regras da mesma. Ademais, contém o manual com especificidades de uso e aplicações que devem ser baixadas para utilizar o analisador léxico da linguagem.

1. Introdução

De modo a considerar as etapas de tradução executadas por um compilador em um código, tem-se: as Análises Léxica, Sintática e Semântica. Este trabalho visa realizar um estudo que tem por base a primeira das três: a Análise Léxica.

Tal análise é realizada pelo analisador léxico do compilador. Este lê a sequência de caracteres, transformando o código-fonte e agrupando os caracteres em sequências com significados, chamadas *lexemes*. Para cada lexeme, o analisador tem como saída um token, sendo que este passa para a fase seguinte de análise: a Sintática. (AHO et. al., 1986). Dito isso, é possível perceber que o analisador tem como função verificar a sintaxe das “palavras” — ou *tokens* — presentes no código-fonte, de modo a garantir que estas estejam escritas corretamente e, portanto, contidas no alfabeto da linguagem.

Dessa maneira, visando exercitar esta etapa de tradução feita pelos compiladores, este trabalho tem a finalidade de apresentar uma linguagem, desenvolvida pelos autores, além de sua análise léxica. Para isso, definiu-se o alfabeto da linguagem, bem como a gramática e as regras de formação para a suas cadeias. Ademais, utilizou-se algumas aplicações que possibilitaram a implementação de um analisador léxico para a linguagem criada.

A linguagem, suas particularidades, gramática e expressões regulares utilizadas serão explicadas nas seções subsequentes. Por fim, este trabalho conterá um manual — documento anexado —, explicando como executar a análise léxica da linguagem proposta.

2. Linguagem

2.1 Alfabeto da linguagem

O alfabeto da linguagem é o conjunto de todos os símbolos que podem compor determinada cadeia de caracteres pertencentes a linguagem em questão. Tal conjunto é composto pelos seguintes elementos:

$$\Sigma = \{\mathbf{a},...,\mathbf{z},\mathbf{0},...,\mathbf{9}, \mathbf{,}, \mathbf{;}, \mathbf{(}, \mathbf{)}, \mathbf{[}, \mathbf{]}, \mathbf{\{}, \mathbf{\}}, /, *, \mathbf{-}, \mathbf{+}, \mathbf{\&}, |, \mathbf{!}, \mathbf{=}, \mathbf{>}, \mathbf{<}\}$$

2.2 Declaração de variáveis

A declaração de variáveis segue o padrão semelhante ao padrão adotado pela linguagem C. Ou seja, não é permitida a declaração com variáveis iniciadas por números ou caracteres especiais, sendo assim, podem começar apenas com letras. Porém, podem conter em sua estrutura quaisquer números ou letras.

2.3 Tipagem de dados

2.3.1 Dados numéricos

Os tipos de dados numéricos reconhecidos são *int* e *float.* De modo que sua estrutura entende quaisquer números formados por dígitos e que não possuam “.” (ponto final) no meio de sua construção como *int* — inteiros — e os que possuem, são tratados como *float* — pontos flutuantes.

2.3.2 Cadeia de caracteres

As cadeias de caracteres — *strings* — são também parte da linguagem. Para reconhecê-las, em suas regras de formação consta que quaisquer caracteres que estejam delimitados por *aspas* — exceto as próprias aspas — serão interpretados como cadeia de caracteres.

2.3.3 Vetores

Além dos tipos supracitados, a linguagem possui, ainda, suporte para vetores, indicados pela palavra reservada *array*. A estrutura dos vetores segue o padrão da linguagem C, isto é, o vetor é declarado a partir de um nome de variável seguido de uma dimensão, sendo esta posta entre colchetes.

2.4 Caracteres de pontuação

Os caracteres utilizados para pontuação são.

- Vírgula (,): sua função é separar declarações.
- Ponto e vírgula (;): sua função é finalizar linhas de comando.
- Parênteses (()): estes símbolos indicam início/fim de comandos e condições.
- Colchetes ([]): são utilizados para indicar a delimitação da dimensão de vetores.
- Chaves ({}): este símbolos indicam início/fim de blocos de comando.
- Barras duplicadas (//): de modo a contemplar a opção de comentar, que devem ser utilizadas no início da frase que o desenvolvedor deseja comentar.

2.5 Operadores

2.5.1 Operadores matemáticos

Os operadores matemáticos definidos para a linguagem em questão são:

- Adição/subtração (+ e - respectivamente).
- Multiplicação/divisão (* e / respectivamente).
- Incremento/decremento (++ e -- respectivamente)
- Atribuição (=).

2.5.2 Operadores lógicos

- And (&&)
- Or (||)
- Not (!);
- Igualdade (==);
- Diferença (!=);
- Maior/menor que (>, < respectivamente)
- Maior/menor ou igual que (>=, <= respectivamente)

2.6 Palavras reservadas

2.6.1 Comandos

Os comandos contidos na linguagem são os que representam sentenças condicionais, como *if*, *then* e *else*, que juntos compõem um comando, os que representam laços de repetição (*loop*), como *for,* que indica início de loop com número de iterações pré-determinado), e *while,* que indica início de loop com número de iterações indeterminado.

2.6.2 Outras

Dentre as palavras reservadas, há também *function*, que é a palavra utilizada para indicar a declaração de uma função, e *null*, que indica valor inexistente, ou seja, nulo.

3. Gramática e expressões regulares

A gramática, responsável por determinar as regras de formação de cadeias e por nortear o funcionamento do analisador léxico, pode ser expressa por:

G = {{A, B, C, D, E, F, G, H}, {a,...,z,0,...,9, ,, ;, (,), [,], {, }, /, *, -, +, &, |, !, =, >, <}, S, P}

As expressões regulares, que são, de fato, as regras de formação de cadeias foram utilizadas de modo a obedecer, como já citado, padrões da linguagem C. São elas:

- **S ⇒ A|B|C|D|E|F|G|H:** Esta expressão não está contida no código, estando presente apenas no documento. Sua função é apenas a de organização, de modo a estabelecer os símbolos não-terminais A, B, C, D, E, F, G e H, que serão derivados em cadeias aceitas pela linguagem. Tais símbolos, bem como as regras de formação das cadeias, estão explicados a seguir.

- **A ⇒ if|then|else|for|while|function:** Expressão regular regular regular utilizada para que o analisador aceite os tokens que representam os comandos da linguagem.

- **B ⇒ \"(\\.|[^"\\])*\"**: Expressão regular regular regular regular utilizada para que o analisador aceite os tokens que representam *strings*.

- **C ⇒ [0-9]+:** Expressão regular regular utilizada para que o analisador aceite os tokens que representam números inteiros.

- **D ⇒ [0-9]+"."[0-9]*:** Expressão regular regular utilizada para que o analisador aceite os tokens que representam números em ponto flutuante, ou seja, com casas decimais.
- **E ⇒ [A-Za-z][A-Za-z0-9]*:** Expressão regular regular utilizada para que o analisador aceite os tokens que representam nomes de variáveis.

- **F ⇒ "int"|"float"|"string"|"array":** Expressão regular regular utilizada para que o analisador aceite os tokens que representam os tipos de dados suportados pela linguagem.

- **G ⇒ "+"|"-"|"*"|"/"|"!="|"++"|"--"|"=="|"="|">="|">"|"<="|"<":** Expressão regular regular utilizada para que o analisador aceite os tokens que representam os operadores matemáticos contidos na linguagem e o operador de atribuição.

- **H ⇒ "{"|"}"|"("|")"|","|"""|";"|":":** Expressão regular regular utilizada para que o analisador aceite os tokens que representam os caracteres utilizados para pontuação.

4. Considerações finais

Após realizada a construção da linguagem, isto é, a definição de todos os seus componentes, e a análise léxica da mesma, é possível perceber a importância desta etapa na tradução do código-fonte, feita pelo compilador.

Uma vez que, sem esta análise, erros de sintaxe, ou declarações inválidas, seriam aceitas pelo compilador e, ainda, não haveria a divisão do código-fonte em *tokens*. Assim, as etapas seguintes seriam prejudicadas e a tradução não seria tão efetiva.

Conclui-se, dessa maneira, que a Análise Léxica é um processo essencial na compilação de um código.

5. Referências

AHO, Alfred V. et al. **Compiladores: Princípios, Técnicas e Ferramentas**. 2. ed. Pearson Education, 2007. 648 p.