Análise Sintática

Humberto Moreira Gonçalves

Outubro 2016

1 Análise Sintática

A gramática é essencial para o compilador, visto que ela é responsável pelo conjunto de regras de produção da linguagem, que descreve como formar a implementação, sendo validada pela análise sintática da linguagem. Portanto, é preciso criar a linguagem utilizando as palavras reservadas e os tokens utilizados anteriormente na análise léxica para montar um código completo. Para esse compilador, deve-se considerar o fato do código possuir, variáveis globais, funções e função principal. O código por possuir qualquer função e qualquer varáveis global e função principal independente da ordem. A tabela a seguir, apresenta a implementação da gramática para a análise sintática:

<programa> ::= <statement> <programa>
| <statement>

<statement> ::= <declaracao_de_funcao>
| <declara_var>

<declaracao_de_funcao> ::= <tipo> IDENTIFICADOR (<declaracao_param>) <sequencia_de_declaracao> FIM
| <tipo> IDENTIFICADOR (<declaracao_param>) FIM
| <tipo> IDENTIFICADOR ( ) FIM
| <tipo> IDENTIFICADOR ( ) <sequencia_de_declaracao> FIM

<declaracao_param> ::= <declaracao_param> VIRGULA <tipo> : IDENTIFICADOR
| <tipo> : IDENTIFICADOR

<sequencia_de_declaracao> ::= <declaracao>
| <declaracao> <sequencia_de_declaracao>

<declaracao> ::= <expressao_condicional>
| <expressao_iteracao>
| <expressao_:=>
| <expressao_leitura>
| <expressao_escreva>
| <declara_var>
| <retorna>
| <chamada_de_funcao>

<expressao_condicional> ::= SE <expressao> ENTAO <sequencia_de_declaracao> SENAO <sequencia_de_declaracao> FIM
| SE <expressao> ENTAO <sequencia_de_declaracao> FIM

<expressao_iteracao> ::= REPITA <sequencia_de_declaracao> ATE <expressao>
<expressao_:=> ::= IDENTIFICADOR := <expressao>
<expressao_leitura ::= LEIA (IDENTIFICADOR)
<expressao_escreva ::= ESCREVE (<expressao>)
<declara_var ::= <tipo> : IDENTIFICADOR

| <tipo> : IDENTIFICADOR VIRGULA <declara_outra_var>
| <tipo> : IDENTIFICADOR := <expressao>
<declara_outra_var> ::= IDENTIFICADOR

<retorna> ::= RETORNA (<expressao>)
<chamada_de_funcao> ::= IDENTIFICADOR (<param_chama_funcao>)

<param_chama_funcao> ::= <param_chama_funcao> VIRGULA <expressao>
| <expressao>
<expressao> ::= <expressao_simples>
| <expressao_simples> <comparacao_operador> <expressao_simples>

<comparacao_operador ::= MAIOR
| MAIORIGUAL
| MENOR
| MENORIGUAL
| IGUALDADE

<expressao_simples> ::= <expressao_simples> <soma> <termo>
| <termo>

<soma> ::= ADICAO
| SUBTRACAO
<mult> ::= MULTIPLICACAO
| DIVISAO
<termo> ::= <fator>
| <termo> <mult> <fator>
<fator> ::= (<expressao>)
| <chamada_de_funcao>
| <expressao_numero>
| <expressao_identificador>
<expressao_identificador> ::= IDENTIFICADOR

<expressao_numero> ::= <numero>

<tipo> ::= INTEIRO
| FLUTUANTE

<numero> ::= INTEIRO
| FLUTUANTE

Como é necessário a criação de um nó da árvore para cada derivação da análise sintática, ao final de cada derivação é adicionado um nó na árvore com a seguinte chamada ArvoreSintatica("tipo", filho, folha). O tipo seria quem produziu, o parâmetro filho é vazio caso a produção não gere nenhuma outro produção, caso contrário é necessário passar a posição da chamada na gramática, e o folha é vazio se a produção não gerar nenhum simbolo terminal, se gerar, é necessário passar a posição do símbolo terminal na produção. O segundo e o terceiro parâmetro é uma lista, portanto é necessário passar os nós filho e folha como lista, ficando da seguinte forma ArvoreSintatica('tipo', [folha], [filho]).