Análise Sintática da linguagem TPP

Henrique S. Marcuzzo[1]

[1]Departamento de Ciência da Computação
– Universidade Tecnológica Federal do Paraná (UTFPR)

henriquemarcuzzo@alunos.utfpr.edu.br

Abstract. *This article describes the steps taken to perform the syntactic analysis work, with a brief explanation of the syntactics adopted, details of how the code was produced, and the experience of working with the technologies used.*

Resumo. *Este artigo descreve os passos tomadas para a execução do trabalho de análise sintática, com um breve explicação sobre sintática adotada, detalhes de como foi produzido o código e a experiência de se trabalhar com as tecnologias usadas.*

1. Introdução

Este trabalho foi realizado com a linguagem de programação *Python* juntamente com a biblioteca *PLY Yacc*, *Anytree* e *Graphviz*, para a Análise sintática dos códigos `.tpp`, durante o processo foi desenvolvido métodos para tratamento de erros sintáticos e aprimorado alguns métodos existentes no código de início, para que atendessem as necessidade descritas na documentação da linguagem de programação fictícia da disciplina **TPP**.

2. Descrição da gramática TPP

A gramática *Tpp*, possui, segundo o padrão *BNF*, as seguintes regras gramaticais sintáticas: A linguagem *Tpp*, foi construída na disciplina de compiladores, com intuito didático, para proporcionar aos alunos uma noção de como funciona a construção de um compilador, portanto a linguagem estudada oferece apenas as estruturas mais simples de uma linguagem convencional, que são:

- programa
  - lista_declaracoes
- lista_declaracoes
  - lista_declaracoes declaracao
  - declaracao
- declaracao
  - declaracao_variaveis
  - inicializacao_variaveis
  - declaracao_funcao
- declaracao_variaveis
  - tipo DOIS_PONTOS lista_variaveis
- inicializacao_variaveis
  - atribuicao
- lista_variaveis
  - lista_variaveis VIRGULA var

    - var
- var
    - ID
    - ID indice
- indice
    - indice ABRE_COLCHETE expressao FECHA_COLCHETE
    - ABRE_COLCHETE expressao FECHA_COLCHETE
- tipo
    - INTEIRO
    - FLUTUANTE
- declaracao_funcao
    - tipo cabecalho
    - cabecalho
- cabecalho
    - ID ABRE_PARENTESE lista_parametros FECHA_PARENTESE
    - corpo FIM
- lista_parametros
    - lista_parametros VIRGULA parametro
    - parametro
    - vazio
- parametro
    - tipo DOIS_PONTOS ID
    - parametro ABRE_COLCHETE FECHA_COLCHETE
- corpo
    - corpo acao
    - vazio
- acao
    - expressao
    - declaracao_variaveis
    - se
    - repita leia
    - escreva
    - retorna
    - erro
- se
    - SE expressao ENTAO corpo FIM
    - SE expressao ENTAO corpo SENAO corpo FIM
- repita
    - REPITA corpo ATE expressao
    - SE expressao ENTAO corpo SENAO corpo FIM
- atribuicao
    - var ATRIBUICAO expressao
- leia
    - LEIA ABRE_PARENTESE var FECHA_PARENTESE
- escreva
    - ESCREVA ABRE_PARENTESE expressao FECHA_PARENTESE
- retorna

    - RETORNA ABRE_PARENTESE expressao FECHA_PARENTESE
- expressao
- expressao$_{l}ogicaatribuicao$
- expressao_logica
    - expressao_simples
    - expressao_logica operador_logico expressao_simples
- expressao_simples
    - expressao_aditiva
    - expressao_simples operador_relacional expressao_aditiva
- expressao_aditiva
    - expressao_multiplicativa
    - expressao_aditiva operador_soma expressao_multiplicativa
- expressao_multiplicativa
    - expressao_unaria
    - expressao_multiplicativa operador_multiplicacao
    - expressao_unaria
- expressao_unaria
    - fator
    - operador_soma fator
    - operador_negacao fator
- operador_relacional
    - MENOR
    - MAIOR
    - IGUAL
    - DIFERENTE
    - MENOR_IGUAL
    - MAIOR_IGUAL
- operador_soma
    - MAIS
    - MENOS
- operador_logico
    - E
    - OU
- operador_negacao
    - NAO
- operador_multiplicacao
    - VEZES
    - DIVIDE
- fator
    - ABRE_PARENTESE expressao FECHA_PARENTESE
    - var
    - chamada_funcao
    - numero
- numero
    - NUM_INTEIRO
    - NUM_PONTO_FLUTUANTE
    - NUM_NOTACAO_CIENTIFICA

- chamada_funcao
  - ID ABRE_PARENTESE lista_argumentos FECHA_PARENTESE
- lista_argumentos
  - lista_argumentos VIRGULA expressao
  - expressao
  - vazio

3. Formato na Análise Sintática

Para o desenvolvimento deste trabalho, foi utilizado a árvore sintática *LALR(1)*, visto que esse tipo de árvores diminui a quantidade de estados e consequentemente o tamanho das tabelas finais em comparação com *LR(1)*, e como a ferramenta escolhida *PLY Yacc*, suporta esse tipo de análise, esta foi a decisão mais sentata, até por questões de desempenho, para o desenvolvimento desta etapa do trabalho.

4. Ferramenta *Yacc*

Para a execução deste trabalho foi escolhido a linguagem de programação *Python* e a biblioteca *PLY Yacc*, assim toda implementação de tratamento para construção da árvore seguiu a lógica determinada pelo *Yacc* que mostrou-se prática e de fácil entendimento.

A base de implementação do analisador sintático foi fornecida pelo professor da disciplina, e ficando a cargo dos alunos aprimorar o que já foi implementado e produzir o código para os tratamentos de eventuais erros sintáticos que podem acontecer durante a análise.

5. Árvore Sintática

Para a construção da *Árvore Sintática*, foi utilizado o código fornecido pelo professor *mytree.py*, que fez uso da biblioteca *anytree* e *graphviz*.

No final do processo, duas árvores são geradas, uma com todos os valores únicos e a outra visualização se faz com todos os operadores sintáticos presentes no código.

6. Referências

Moodle. Gramática da Linguagem T++ com TOKENS (para a lista de tokens). 2021. Disponível em: *https://docs.google.com/document/d/1e7_M-bD1RUbJAnyR8rZyJ35vKbYEN6KQG4l5L8FQ7_I/edit*. Acesso em: 01 abr.2021.