Análise Semântica 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 semantic analysis work, with a brief explanation of the strategy adopted, details of how the code was produced, and a basic example output.*

Resumo. *Este artigo descreve os passos tomadas para a execução do trabalho de análise semântica, com um breve explicação sobre a estratégia adotada, detalhes de como foi produzido o código e um exemplo básico de saída.*

1. Introdução

Este trabalho foi realizado com a linguagem de programação *Python* juntamente com a biblioteca *PLY Yacc*, *Anytree* e *Graphviz* e *Tabulate*, para a Análise semântica dos códigos `.tpp`, durante o processo foi desenvolvido métodos para criação de tabelas de funções e variáveis e a partir dessa tabela foram feito todas as checagem descitas na documentação da linguagem forneceido pelo professor.

2. Análise sensível ao contexto

Para fazer a análise semântica da linguagem, foi definido uma estrutura para tabela de variáveis e de função, sendo possível assim fazer todas as checagem utilizando apenas as tabelas, os campos para tabela de variáveis e função podem ser visto na imagem 1 e 2 respectivamente.

Nome	Tipo	Dimensões	Tamanho Dimensões	Escopo	Linha(s) declarada	Linha(s) chamadas

Figura 1. Tabela de variáveis

Nome	Tipo	Num. Parâmetros	Parâmetros	Retorno	Linha declarada	Linha finalizada	Declarada	Linha(s) chamadas

Figura 2. Tabela de funções

Para a tabela de variáveis os campos tem por finalidade:

- **Nome**: O nome da variável.
- **Tipo**: O tipo da variável (inteiro ou flutuante).
- **Dimensões**: Quantas dimensões a variável possui, sendo 0 para caso não tenha .

- **Tamanho Dimensões**: Uma lista com o tamanho máximo de cada dimensão que possuir, sendo uma lista vazia caso não possua dimensões.
- **Escopo**: A qual escopo ela pertence, tendo por valor padrão o escopo ”global”.
- **Linha(s) declaradas**: Em qual linha a variável foi declarada, podendo ser mais de uma vez no mesmo escopo ou em escopos diferentes.
- **Linhas(s) chamadas**: Uma lista de tupla com as linha em que foi chamada e o seu nó respectivo da árvore.

Para a tabela de funções os campos tem por finalidade:

- **Nome**: O nome da função.
- **Tipo**: O tipo da função (inteiro ou flutuante).
- **Num. Parâmetros**: Quantos parâmetros a função possui, sendo 0 (zero) se não possuir parâmetro.
- **Parâmetros**: Uma lista com o nome de cada variável de parâmetro, sendo uma lista vazia caso não possua dimensões.
- **Retorno**: Uma lista com todos os tipos dos possiveis retorno da função.
- **Linha declaradas**: Em qual linha a função foi declarada.
- **Linha finalizada**: Em qual linha a função foi finalizada.
- **Declarada**: Campo de controle, obtendo o valor ”True”caso a função tenha sido declarada e ”False”caso contrario (Necessário para o caso das funções recursivas).
- **Linhas(s) chamadas**: Uma lista de tupla com as linha em que foi chamada e o seu nó respectivo da árvore.

Com essas duas tabelas formadas foi possível fazer todas as checagem semânticas necessárias para a linguagem tpp, se existe função principal, se os retornos das funções estão de acordo ao seu tipo, se as chamadas de funções estão sendo passado todos os campos necessários, checar se as várias que estão sendo chamadas existem, o escopo a que pertencem, se na atribuição o tipo da expressão está de acordo com o que a variável espera, se os acessos em variáveis de array estão dentro do seu espaço correto e se são números inteiros.

Entre outros como checar se uma função já foi declarada uma vez, que no caso da função isso não era permitido, e se a variável já foi declarada mais de uma vez (no mesmo escopo ou em escopo diferente).

3. Geração da tabela de símbolos

Para gerar as tabelas 1 e 2, foi utilizado a estrutura do código da análise sintática, sendo assim, a tabela foi sendo gerada juntamente com a árvore, percorrendo a árvore a partir de nós ”chaves”, sendo eles os nós de *cabecalho*, *declaracao_variavel*, *retorna*, *leia*, *escreva*, *atribuicao* e *chamada_funcao*.

Praticamente todos os atributos da tabela puderam ser gerados a partir do código sintático, exceto o escopo das variáveis que eram declaradas dentro de uma função, estes tiveram que ser ajustados na análise semântica visto que devido a árvore ser montada de baixo para cima, no momento em que estas variáveis eram declaradas e chamadas, não era possível determinar se estava dentro de uma função ou não, logo as chamadas destas variáveis também foram ajustados após sua formação.

A estratégia utilizado no campo de *Linha(s) chamadas*, onde foi feito uma tupla com a linha onde a variável ou função foi chamada e o seu respectivo nó, se da devido ao fato de que seria necessário fazer checagens nestas chamadas, como por exemplo se todos os parâmetros necessários para função estavam sendo enviado.

4. Exemplo de entrada e saída

Para cada código *.tpp* submetido a está parte do projeto, a análise semântica irá gerar uma tabela de funções e uma de variável, como já foi descrito anteriormente e irá gerar uma árvore simplificada da árvore gerada pela análise sintática.

Podemos vem um simples exemplo de saída quando executamos o código abaixo:

```
inteiro: a
flutuante: b
inteiro: c[1.2]

inteiro principal()
  c[5.8] := 10

fim
```

A tabela de funções e variáveis pode ser vista na imagem 3, erros e avisos do exemplo estão na imagem 4, e a árvore simplificada na imagem 5.

TABELA DE FUNÇÕES:

Nome	Tipo	Num. Parâmetros	Parâmetros	Init	Linha Inicial	Linha Final
principal	inteiro	0	[]	True	10	13

TABELA DE VARIÁVEIS:

Nome	Tipo	Dimensões	Tamanho Dimensões	Escopo	Linha
a	inteiro	0	[]	global	6
b	flutuante	0	[]	global	7
c	inteiro	1	[1.2]	global	8

Figura 3. Tabela de saída (Função e Variável

```
Erro: Índice de array 'c' não inteiro.
Aviso: Variável 'a' declarada e não inicializada.
Aviso: Variável 'b' declarada e não inicializada.
Erro: Função principal deveria retornar inteiro, mas retorna vazio.
```

Figura 4. Avisos e erros de saída da análise sintática

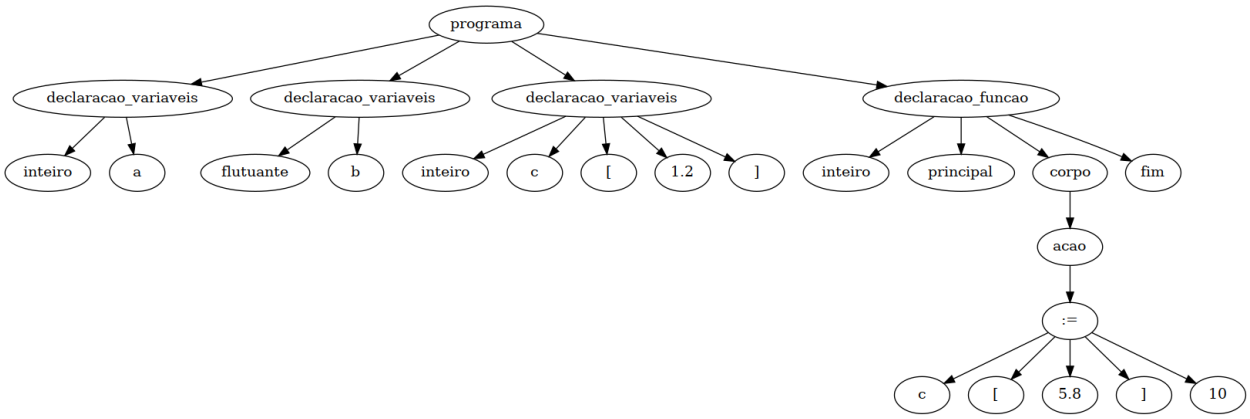

Figura 5. Arvore Simplificada

5. Referências

Moodle. Projeto de Implementação de um Compilador para a Linguagem T++. 2021. Disponível em: *https://moodle.utfpr.edu.br/pluginfile.php/183223/mod_resource/content/13/trabalho-03.md.notes.pdf*. Acesso em: 23 abr. 2021.