Projeto de Compiladores - Etapa 5

Tabela de Símbolos e Análise Semântica

Artur Levy Lima Barbosa
Déborah Abreu Sales
Gabriel Miranda Oliva
Rafael Takeguma Goto

Aracaju, Sergipe
2024

1. Classes Java

As duas principais classes implementadas foram Semantico.java e SymbolTableManager.java, onde a classe Semantico.java foi responsável por toda lógica da análise semântica e a SymbolTableManager.java se responsabilizou em “administrar” a tabela de símbolos, por meio de uma pilha de CustomHashMaps.

1.1. Semantico.java

Inicialmente, a classe herda de DepthFirstAdapter e utiliza o padrão de visitas para percorrer a árvore sintática gerada na análise. A estrutura central do código é a gestão de símbolos e escopos, realizada pela classe auxiliar SymbolTableManager. Durante o processo de análise, cada vez que um bloco de código é iniciado ou terminado (como funções e loops), a tabela de símbolos entra ou sai de um novo escopo. Isso garante que variáveis e funções declaradas sejam corretamente registradas e validadas em seus respectivos escopos.

Além disso, a classe utiliza uma pilha (typeStack) para verificar a compatibilidade de tipos de variáveis e expressões, empurrando e retirando os tipos conforme operações ou atribuições são encontradas no código. Quando ocorre uma operação binária, como soma ou comparação, a classe verifica se os tipos são compatíveis e se os operandos são válidos.

A classe verifica se os tipos das operações são compatíveis com o que é permitido, o método “binaryOperation” é utilizado para os métodos das operações, ele pega os dois elementos do topo de typeStack e confere se eles são tipos permitidos para a determinada operação.

Além de verificar as operações e atribuições, a classe também emite erros detalhados caso haja problemas, como variáveis não declaradas, tipos incompatíveis ou vetores mal utilizados, tornando o processo de programação e compilação em caju mais amigável ao usuário.

1.2. SymbolTableManager.java

Possui o atributo “tableStack” que é uma pilha de tabelas hash customizadas. onde cada tabela representa um novo escopo, ou seja, sempre que o método “enterScope” é chamado uma nova tabela é adicionada ao topo da pilha, e sempre que um escopo termina é chamado o método “exitScope” que remove a tabela que se encontra no topo da “tableStack”.

Ademais, esta classe possui os métodos básicos de adicionar um novo símbolo: “addSymbol” e de buscar um símbolo pelo nome: “lookup”.
Dentro desta classe existem 3 classes internas: Symbol, que define um símbolo com um tipo e valor, (2) CustomHashMap que é o hashMap personalizado para armazenar os símbolos e (3) Entry que está dentro de CustomHashMap e representa uma entrada da tabela hash.

2. Modo de Uso

Em Main.java substituir a String “arquivo” pelo caminho do arquivo com extensão .cj que se deseja trabalhar. Executar a classe Main.java.