Departamento de Ciência da Computação - UFJF
DCC045 - Teoria dos compiladores

Geração de Código para a Linguagem *lang*

Alunos:

Daniel Souza Ferreira - 201665519B

Matheus de Oliveira Carvalho - 201665568C

Setembro/ 2021

1 Introdução

Neste trabalho desenvolvemos o gerador de código para finalizar a criação de um compilador para a linguagem *"lang"*.

Após verificar que um código é lexicamente, sintaticamente e semanticamente correto a etapa final do compilador consiste na geração do código objeto. Para uma arquitetura real o produto gerado é em código de máquina, porém nesse trabalho a geração do código foi feita para Java, sendo a mesma uma linguagem de alto nível.

2 Características do Projeto

Para o desenvolvimento da parte final do compilador foi utilizada a biblioteca para templates StringTemplate disponibilizado no site stringtemplate.org em sua versão 4.3.1 por meio da adição do arquivo ***ST-4.3.1.jar*** e as técnicas de criação de template e de um visitor para essa estrutura, mostradas pelo professor nas aulas e nas documentações encontradas no site do StringTemplate.

O template criado para a geração do código em Java e o visitor responsável pelo preenchimento do template com valores para criação do código java estão presentes, respectivamente, nos arquivos **java.stg** e **VisitorJava.java**. Para completa realização da geração foi criada 2 funções novas, **getCompilersFuncs e getCompilersEnvs** no arquivo VisitorSemantic (arquivo utilizado no trabalho anterior) que foi renomeado para esse trabalho como **VisitorCompiler**. Esses novos métodos são responsáveis por retornar as variáveis *compilersEnvs* e *compilersFuncs* que representam os escopos das funções em ordem de chamadas e o tipo de retorno delas quando as mesmas são chamadas como expressão. Um novo arquivo para testes **TestVisitorCompiler** foi criado seguindo o princípio dos outros arquivos de testes usados mas dessas vez utilizando dois Visitor: o primeiro é o VisitorCompiler, que fará a análise semântica do código lang e depois o VisitorJava que irá gerar o código Java com a ajuda do Template.

2.1 Novos Arquivos Presentes no Trabalho

- ST-4.3.1.jar
- java.stg
- VisitorJava.java
- VisitorCompiler.java

- TestVisitorCompiler.java

3 Decisões de Projeto

3.1 Estruturas

- **groupTemplate:** Uma váriavel do tipo STGroup responsável por inicializar o template criado.
- **type, stmt, expr:** Variáveis do tipo ST responsáveis por representar tipos, comandos e expressões presentes no código.
- **funcs, params, datas, decls:** Lista de variável do tipo ST responsáveis por representar funções, parámetros, datas e declarações presentes no código.
- **fileName:** Uma variável do tipo String que representa o nome do arquivo a ser gerado.
- **file:** Uma variável do tipo FileWriter que representa o arquivo onde o código será gerado.
- **localEnv:** Hashmap para armazenar os tipos de variáveis. Contém Strings representado o id do tipo funcionando como key e variáveis do tipo SType funcionando como value.
- **env:** Uma lista de HashMap para armazenar os escopos das funções em ordem de chamada delas.
- **functions:** Uma lista de STypeFunc usada para determinar quais são os tipos de retornos de funções que são chamadas como expressões.

3.2 Decisões

- O programa se encerra ao encontrar um erro.
- Na operação **Iterate** utilizamos uma técnica para evitar que a variáveis do for sejam repitidas armazenando nelas o valor da linha e coluna no código lang onde ocorre a operação de interação
- Para a operação **New** utilizamos 4 tipos diferentes para cada tipo de contexto.
- Foi desenvolvido metodos auxiliares para obtenção de valores 'default' para os Types e STypes.

4 Execução do Programa

Para a execução do programa foi criado um makefile que possui os seguintes comandos:

- **make** - utilizado para fazer a limpeza dos arquivos .class gerados na pasta 'lang' e depois realizar a compilação de todos os arquivos .java.
- **make clean** - utilizado para fazer a limpeza dos arquivos .class gerados na pasta 'lang'.
- **make execute** - utilizado para realizar a execução da bateria de testes.

A bateria de testes gerada pelo make execute é dos arquivos da pasta 'certa' da pasta 'semantica'. Para realizar os testes dos arquivos da pasta 'errado' deve-se alterar o valor da variável **okSrcs** na classe **TestVisitorCompiler.java** para **"testes/semantica/errado/"**.

Nessa parte do trabalho os arquivos do make sofreram maiores alterações em relação as mudanças anteriores, sendo necessário a inclusão de ***:ST-4.3.1.jar*** na linha de comnado de compilação e execução uma vez que esse comando é necessario para usar o StringTemplate.