UNIVERSIDADE DO VALE DO RIO DOS SINOS

Exercício 3

Nomes: Bruno da Siqueira Hoffmann e Rafael Hansen Klauck.
Atividade Acadêmica: Compiladores.
Ano/Semestre: 2024/02.
Professor: Flávio Oliveira

Exercício 3 - analisador sintático e gerador
de código para uma linguagem de programação

Informações Gerais: Com o intuito de gerar os comandos (TAC) e a tabela de símbolos, foi necessário fazer algumas adaptações em nosso antigo trabalho, adicionando uma classe capaz de verificar as chamadas de cada regra de produção e possibilitar a adição um ação ao acontecer essa regra. Dessa forma, foi possível gerar o código intermediário e a tabela de símbolos. Com relação aos arquivos presentes na entrega é possível visualizar:

- **entrega_brunoh_rafaelk.zip:** contendo os arquivos presentes no repositório que trabalhamos.

A estrutura de pastas que você irá verificar ao fazer o *unzip* do arquivo será:

- **README.md:** arquivo contendo informações de como executar o gerador de comandos TACs e tabela de símbolos.
- **/src**: pasta contendo todas as classes java, como também a classe com a gramática (FoolGrammar.g4).
    - **FOOLISemanticoListener.java**: arquivo contendo o código necessário para geração do código intermediário;
- **/tests:** os arquivos de *input* e *output* que utilizamos para testar;
    - **input.txt:** arquivo de teste de entrada utilizado, contendo o código FOOLI;
    - **output.txt:** arquivo de teste de saída, contendo a TAC e tabela de símbolos.

Repositório Git: Além disso, é possível acompanhar nosso desenvolvimento no nosso repositório Git (https://github.com/BrunoHoffmann15/trab-compiladores-ga-2). Obs: mantivemos no repositório da última entrega.

Teste: Para testar a geração dos Comandos (TACs) foi criado o código FOOLI contendo o método main, métodos auxiliares, chamadas entre métodos, mais de uma operação em uma única linha, estrutura de repetição e estruturas condicionais. Abaixo é possível verificar o código FOOLI gerado para teste, sendo ele salvo em um arquivo chamado *input.txt.*

```
class Main {
  int val;
  int valDois;

  void main() {
    int valorIncrementado;
    bool condicional;
    valorIncrementado = funcao2(val);

    condicional = valorIncrementado < valorIncrementado;
    if (condicional) return val + valDois; else return valDois;;
  }

  int funcao2(int value) {
    return value + 1;
  }

  int funcao() {
    while (val < valDois) val = val + 1;
    int valTres;
    valTres = val + valDois + 1;
    int soma;
    soma = valTres + valDois;
    return soma;
  }
}
```

Em seguida, foi executado os comandos necessário para geração dos Comandos (TACs) e da tabela de símbolos, dessa forma foi executado o comando **java -cp "../tools/antlr-4.13.2-complete.jar:." Main input.txt output.txt** e foi possível ter como saída o arquivo output, contendo as informações:

```
================================
Tabela de Símbolos:
================================
val: int
valDois: int
valTres: int
soma: int
condicional: bool
value: int
```

```
valorIncrementado: int

==============================
Código Intermediário (TAC):
==============================
main:
param val
t0 = call funcao2, 1
valorIncrementado = t0
t1 = valorIncrementado < valorIncrementado
condicional = t1
if not(condicional) goto L0
t2 = val + valDois
return t2
goto L1
L0:
return valDois
L1:

funcao2:
t3 = value + 1
return t3

funcao:
L2:
t4 = val < valDois
if not(t4) goto L3
t5 = val + 1
val = t5
goto L2
L3:
t6 = val + valDois
t7 = t6 + 1
valTres = t7
t8 = valTres + valDois
soma = t8
return soma
```