Universidade Estadual do Oeste do Paraná - UNIOESTE
Centro de Engenharias e Ciências Exatas - CECE
Campus de Foz do Iguaçu

Isabela Pimentel Loebel
Vivian Miwa Fugihara

TRABALHO 1 DE COMPILADORES
ANALISADOR LÉXICO - CTRL C

FOZ DO IGUAÇU
2022

Sumário

Objetivo e Funcionamento do Programa

Neste relatório serão apresentadas as ferramentas e os métodos utilizados para a resolução do primeiro trabalho da disciplina de compiladores. Para o seu desenvolvimento, foi utilizado o software Eclipse IDE na versão 4.25.0.

A linguagem Ctrl C, escrita na linguagem Java, foi construída com o apoio de uma ferramenta para criação de compiladores, o JavaCC, que gera a análise léxica. Essa ferramenta reconhece um subconjunto da linguagem C e foi utilizada sua versão mais recente, a 7.0.10.

O programa é compilado pelo JavaCC e executado na própria IDE do Eclipse, e após ser executado, é preciso entrar com o nome do arquivo que contém o código a ser analisado. Esse arquivo deve ser um arquivo texto, com a notação “nomeDoArquivo.txt”.

Caso haja necessidade do analisador léxico Ctrl C ser executado em outra plataforma, como por exemplo o próprio prompt de comando, deve-se abrir o diretório onde é armazenado os arquivos (deve conter todos eles, inclusive a pasta Tables) e executar o comando “javacc CtrlC.jj”. Em seguida, o comando “javac *.java”, e por fim, “java CtrlC”. É necessário ressaltar que o projeto foi desenvolvido no Eclipse IDE, sendo assim, o java JDK e o java JRE instalados em seu computador devem ser coerentes para que a execução pelo prompt seja realizada. Caso contrário, apenas funcionará no Eclipse com java JDK versão 19.0.1.

No arquivo "CtrlC.jj" estão dispostos todos os tokens e suas regras, assim como a classe CtrlC, que faz a declaração de cada token. Desse modo, a linguagem Ctrl C reconhece as seguintes classes de tokens, métodos e regras:

Classes de Tokens

- Operadores:
    - < SOMA : "+" >
    - < SUB : "-" >
    - < MULT : "*" >
    - < DIV : "/" >
    - < MOD : "%" >
    - < ATRIBUICAO : "=" >
    - < AND : "&&" >
    - < OR : "||" >
    - < EQUAL : "==" >
    - < DIFFERENT : "!=" >
    - < MAIOR : ">" >
    - < MENOR : "<" >
    - < MAIORIGUAL : ">=" >
    - < MENORIGUAL : "<=" >
    - < ATRIBUISOMA : "+=" >
    - < ATRIBUISUB : "-=" >
    - < INC : "++" >
    - < DEC : "--" >
- Símbolos:
    - < PARENTHESESO : "(" >
    - < PARENTHESESC : ")" >

    - < CHAVESA : "{" >
    - < CHAVESF : "}" >
    - < PONTOVIR : ";" >
- Palavras reservadas:
    - < MAIN : "main" >
    - < IF : "if" >
    - < ELSE : "else" >
    - < INT : "int" >
    - < FLOAT : "float" >
    - < DOUBLE : "double" >
    - < CHAR : "char" >
    - < PRINT : "print" >
    - < FOR : "for" >
    - < WHILE : "while" >
    - < VOID : "void" >
    - < RETURN : "return" >
- Identificadores:
    - < NUM :(["0"-"9"])+(["."] (["0"-"9"])+)? >
    - < ID : ["a"-"z", "A"-"Z"] (["a"-"z", "A"-"Z", "0"-"9", "_"])* >

Métodos e regras

- **public static void main (String args[])**: método responsável pela busca e leitura do arquivo de entrada, com o auxílio das bibliotecas Java.io e Scanner, próprias da linguagem Java;
- **static final public void iniciar()**: método responsável por fazer a inicialização do analisador e a criação da tabela de símbolos e de palavras reservadas;
- **SymbolTable**: classe responsável por estruturar a tabela de símbolos e de palavras reservadas, que é implementada com uma lista SymbolTable. A classe possui os seguintes atributos:
    - symbol: caractere;
    - tok: token;
    - categoria: categoria da variável;
    - type: tipo da variável.
- **SKIP**: classe responsável por indicar comandos que devem ser ignorados na leitura dos tokens:
    - “ ”
    - "\n”
    - "\t"
    - "\r\n"
    - "\r"
- **TOKENS**: classe responsável pela definição dos tokens para análise. Esses tokens podem ser divididos em:
    - Símbolos:
    - Palavras reservadas;
    - Operadores;
    - Identificadores.

Identificação de Erros

O programa, ao se deparar com um erro léxico, é interrompido e identifica os seguintes erros, assim como sua posição:

- Números decimais com “,” (vírgula) em vez de “.” (ponto);
- Variáveis nomeadas com caracteres especiais antes e entre as letras, como “@b” e “tes@te”.

Referências

1. JavaCC. (2022). Documentação JavaCC. <https://javacc.github.io/javacc/documentation/jjdoc.html>. Acesso em 14 de novembro de 2022.
2. Bordini, Camile. (2022). Aula 2: Análise Léxica.
3. Bordini, Camile. (2022). Aula 3: Tabela de Símbolos.