Universidade Federal de Ouro Preto
Instituto de Ciências Exatas e Biológicas
Departamento de Computação
BCC - 328 Construção de Compiladores I

Trabalho Prático 1

Análisador léxico para a linguagem de programação de Torben

Aluno: Lucas de Rocha Castro - 14.1.4210
Professor: Dr. José Romildo Malaquias

1- Linguagem Torben

A linguagem Torben é apresentada em seu livro da seguinte maneira:

$$
\begin{array}{lll}
\textit{Program} & \rightarrow & \textit{Funs} \\
 & & \\
\textit{Funs} & \rightarrow & \textit{Fun} \\
\textit{Funs} & \rightarrow & \textit{Fun Funs} \\
 & & \\
\textit{Fun} & \rightarrow & \textit{TypeId} \; (\; \textit{TypeIds} \;) = \textit{Exp} \\
 & & \\
\textit{TypeId} & \rightarrow & \texttt{int} \;\; \textbf{id} \\
\textit{TypeId} & \rightarrow & \texttt{bool} \;\; \textbf{id} \\
 & & \\
\textit{TypeIds} & \rightarrow & \textit{TypeId} \\
\textit{TypeIds} & \rightarrow & \textit{TypeId} \; , \;\; \textit{TypeIds} \\
 & & \\
\textit{Exp} & \rightarrow & \textbf{num} \\
\textit{Exp} & \rightarrow & \textbf{id} \\
\textit{Exp} & \rightarrow & \textit{Exp} + \textit{Exp} \\
\textit{Exp} & \rightarrow & \textit{Exp} < \textit{Exp} \\
\textit{Exp} & \rightarrow & \texttt{if} \; \textit{Exp} \; \texttt{then} \; \textit{Exp} \; \texttt{else} \; \textit{Exp} \\
\textit{Exp} & \rightarrow & \textbf{id} \; (\; \textit{Exps} \;) \\
\textit{Exp} & \rightarrow & \texttt{let} \; \textbf{id} = \textit{Exp} \; \texttt{in} \; \textit{Exp} \\
 & & \\
\textit{Exps} & \rightarrow & \textit{Exp} \\
\textit{Exps} & \rightarrow & \textit{Exp}, \;\; \textit{Exps}
\end{array}
$$

2- Desenvolvimento

O análisador léxico foi desenvolvido utilizando a ferramenta *J-flex*, que cria uma classe *Java* que realiza a análise léxica de qualquer entrada em forma de texto. A criação do análisador léxico pelo *J-Flex* só é possível se o informarmos as regras que compõe essa classe.

Foi definido os tokens que representação a atribuição, igualdade, abre e fecha parênteses, operadores lógicos, tipos, etc… como mostra na imagem abaixo:

```
":="            { return tok(ASSIGN); }
"="             { return tok(EQ); }
"("             { return tok(LPAREN); }
")"             { return tok(RPAREN); }
","             { return tok(COMMA); }
"+"             { return tok(PLUS); }
"-"             { return tok(MINUS); }
"*"             { return tok(TIMES); }
"/"             { return tok(DIV); }
"%"             { return tok(MOD); }
"<>"            { return tok(NE); }
"<"             { return tok(LT); }
"<="            { return tok(LE); }
">"             { return tok(GT); }
">="            { return tok(GE); }
"&&"            { return tok(AND); }
"||"            { return tok(OR); }


bool            {return tok(BOOL); }
int             {return tok(INT); }
string          { return tok(STRING); }
if              { return tok(IF); }
then            { return tok(THEN); }
else            { return tok(ELSE); }
while           { return tok(WHILE); }
do              { return tok(DO); }
let             { return tok(LET); }
in              { return tok(IN); }

{id}            { return tok(ID, yytext());}
```

Os comentários foram definidos como:

```
<COMMENT> {
"{#"              { commentLevel --;
                    if (commentLevel == 0)
                        yybegin(YYINITIAL);
                  }
"#}"              { commentLevel ++; }
<<EOF>>           { yybegin(YYINITIAL);
                    System.err.println("error: unclosed
comment");
                  }
```

```
[^]                 {  }
}
```

que começa com # e se não terminar com #, exibe erro.