Yuri Kayser da Rosa 12201035
Alexandre Back 18100844

Relatório de implementação do Analisador Léxico

Iniciamos a implementação do analisador léxico através da instalação do plugin de JavaCC para a IDE Eclipse. Depois, seguimos a implementação básica descrita no livro “Como construir um compilador usando ferramentas Java”, que consistia na definição de alguns tokens de operadores, comentários, String, constantes, palavras reservadas e tratamento de erros.

Depois, iniciamos a implementação dos requisitos descritos no item 5 da tarefa. Segue abaixo a implementação:

a. Float

```
< float_constant:(
((["0"-"9"])+ ["."] (["0"-"9"])+ ) )>
```

b. Booleanos

```
< boolean_constant:( "true" | "false" )>
< BOOLEAN : "boolean" > (definido como palavra reservada)
```

c. Char

```
<  char_constant: // constante char
   "\'" (~["\'","\r","\n"]) "\'"
 | "\'" (("\\""n")) "\'"
 | "\'" (("\\""r")) "\'" >
```

Tratamento de erros Char

```
<INVALID_CHAR:
  "\'" (~ ["\n", "\r","\'"])* ["\n","\r"]>
  {
System.err.println("Line " + input_stream.getEndLine() +  " - Char constant has a \\r: " +
image);
    countLexError++;
System.err.println("Quantidade de erros encontrados ate o momento: " +
countLexError);
  }
|
<EMPTY_CHAR:
  "\'" "\'">
  {
    System.err.println("Line " + input_stream.getEndLine() +  " - Char vazio: " + image);
    countLexError++;
    System.err.println("Quantidade de erros encontrados ate o momento: " +
countLexError);
  }
```

D. Xor, or, And, not

Adicionados como operadores.

< NOT : "not" >

< AND : "and" >

< OR : "or" >

< XOR : "xor" >

A maior dificuldade encontrada foi na parte de implementação de um Char e tratamento de erros deste tipo de dado. Isso ocorreu porque um char deve aceitar ‘\n’ e ‘\r’ mas não pode ter tamanho maior que 1. Por exemplo, um char ‘aa’ não é um char válido. Acabamos não encontrando uma forma de tratar este tipo de erro, entretanto apesar de gerar uma exceção ao ler estas entradas, o analisados não os reconhece com um char, que é o comportamento correto.

Tabela de testes

Token	Resultado
0	Reconheceu int_constant
10	Reconheceu int_constant
129	Reconheceu int_constant
410	Reconheceu int_constant
0.12	Reconheceu float_constant
1.1	Reconheceu float_constant
98.87	Reconheceu float_constant
98.873	Reconheceu float_constant
'a'	Reconheceu char_constant
'\n'	Reconheceu char_constant
'\r'	Reconheceu char_constant
'_'	Reconheceu char_constant
'B'	Reconheceu char_constant
'4'	Reconheceu char_constant
string 1	Reconheceu string_constant
1 CEBRUtius	Reconheceu string_constant
blu3t00th	Reconheceu string_constant
/* asdf	SKIP
comment	SKIP

*/	SKIP
//comm	SKIP
Identifer2	Reconheceu IDENT
break	Reconheceu BREAK
class	Reconheceu CLASS
constructor	Reconheceu CONSTRUCTOR
else	Reconheceu ELSE
extends	Reconheceu EXTENDS
for	Reconheceu FOR
if	Reconheceu IF
int	Reconheceu INT
float	Reconheceu FLOAT
new	Reconheceu NEW
print	Reconheceu PRINT
read	Reconheceu READ
return	Reconheceu RETURN
string	Reconheceu STRING
char	Reconheceu CHAR
super	Reconheceu SUPER
boolean	Reconheceu BOOLEAN
=	Reconheceu ASSIGN
>	Reconheceu GT
<	Reconheceu LT
==	Reconheceu EQ
<=	Reconheceu LE
>=	Reconheceu GE
!=	Reconheceu NEQ
+	Reconheceu PLUS
-	Reconheceu MINUS
*	Reconheceu STAR
/	Reconheceu SLASH
%	Reconheceu REM
not	Reconheceu NOT
and	Reconheceu AND

or	Reconheceu OR
xor	Reconheceu XOR
(	Reconheceu LPAREN
)	Reconheceu RPAREN
{	Reconheceu LBRACE
}	Reconheceu RBRACE
[	Reconheceu LBRACKET
]	Reconheceu RBRACKET
;	Reconheceu SEMICOLON
,	Reconheceu COMMA
.	Reconheceu DOT
'a	Erro
.12	DOT int constant
23.	int constant DOT
"a	erro
a2a	ident
a.28	ident
XOR	ident
""	erro vazio e erro quebra de linha
AND	ident
OR	ident
"	erro
&	erro
|	erro
^	erro
`	erro