Project One : Lexer Output

Jordan Murray

Jordan.Murray1@Marist.edu

March 9, 2021

1 Test Case 1

1.1 Program

{}$

1.2 output

INFO LEXER —> Lexing Program 1
DEBUG LEXER —> R_BRACE [{] Found at (1 : 1)
DEBUG LEXER —> L_BRACE [}] Found at (1 : 2)
DEBUG LEXER —> T_EOP [$] Found at (1 : 3)
INFO LEXER —> Lex completed with 0 errors
INFO LEXER —> Lex completed with 0 warnings

1.3 comments

This is a simple test program that completes with no errors or warnings

2 Test Case 2

2.1 Program

{{{{{{}}}}}}$

2.2 output

```
INFO LEXER —> Lexing Program 2
DEBUG LEXER —> R_BRACE [ { ] Found at ( 3 : 1 )
DEBUG LEXER —> R_BRACE [ { ] Found at ( 3 : 2 )
DEBUG LEXER —> R_BRACE [ { ] Found at ( 3 : 3 )
DEBUG LEXER —> R_BRACE [ { ] Found at ( 3 : 4 )
DEBUG LEXER —> R_BRACE [ { ] Found at ( 3 : 5 )
DEBUG LEXER —> R_BRACE [ { ] Found at ( 3 : 6 )
DEBUG LEXER —> L_BRACE [ } ] Found at ( 3 : 7 )
DEBUG LEXER —> L_BRACE [ } ] Found at ( 3 : 8 )
DEBUG LEXER —> L_BRACE [ } ] Found at ( 3 : 9 )
DEBUG LEXER —> L_BRACE [ } ] Found at ( 3 : 10 )
DEBUG LEXER —> L_BRACE [ } ] Found at ( 3 : 11 )
DEBUG LEXER —> L_BRACE [ } ] Found at ( 3 : 12 )
DEBUG LEXER —> T_EOP [ $ ] Found at ( 3 : 13 )
INFO LEXER —> Lex completed with 0 errors
INFO LEXER —> Lex completed with 0 warnings
```

2.3 comments

This program tests the case of multiple symbols and its ability to keep track of position.

3 Test Case 3

3.1 Program

```
{
boolean a
"a is false
}$
```

3.2 output

```
INFO LEXER —> Lexing Program 3
DEBUG LEXER —> R_BRACE [ { ] Found at ( 5 : 1 )
DEBUG LEXER —> T_BOOL [ boolean ] Found at ( 6 : 3 )
DEBUG LEXER —> T_ID [ a ] Found at ( 6 : 11 )
DEBUG LEXER —> T_QUOTE [ " ] Found at ( 7 : 4 )
DEBUG LEXER —> T_CHAR [ a ] Found at ( 7 : 5 )
DEBUG LEXER —> T_CHAR [ ] Found at ( 7 : 5 )
DEBUG LEXER —> T_CHAR [ i ] Found at ( 7 : 6 )
DEBUG LEXER —> T_CHAR [ s ] Found at ( 7 : 7 )
DEBUG LEXER —> T_CHAR [ ] Found at ( 7 : 8 )
DEBUG LEXER —> T_CHAR [ f ] Found at ( 7 : 9 )
DEBUG LEXER —> T_CHAR [ a ] Found at ( 7 : 10 )
DEBUG LEXER —> T_CHAR [ l ] Found at ( 7 : 11 )
DEBUG LEXER —> T_CHAR [ s ] Found at ( 7 : 12 )
DEBUG LEXER —> T_CHAR [ e ] Found at ( 7 : 13 )
Error: (8:0) Unrecognized Token: }
```

```
DEBUG LEXER —> L_BRACE [ } ] Found at ( 8 : 1 )
Error: (8:1) Unrecognized Token: $
DEBUG LEXER —> T_EOP [ $ ] Found at ( 8 : 2 )
Warning: String did not close
INFO LEXER —> Lex completed with 2 errors
INFO LEXER —> Lex completed with 1 warnings
```

3.3 COMMENTS

This program represents the errors/warnings thrown when a string is not closed and it finds anything other than a charlist, a space, or nothing in a string.

4 Test Case 4

4.1 Program

```
{
"bad str1ng"
}$
```

4.2 OUTPUT

```
INFO LEXER —> Lexing Program 4
DEBUG LEXER —> R_BRACE [ { ] Found at ( 10 : 1 )
DEBUG LEXER —> T_QUOTE [ " ] Found at ( 11 : 3 )
DEBUG LEXER —> T_CHAR [ b ] Found at ( 11 : 4 )
DEBUG LEXER —> T_CHAR [ a ] Found at ( 11 : 4 )
DEBUG LEXER —> T_CHAR [ d ] Found at ( 11 : 5 )
DEBUG LEXER —> T_CHAR [ ] Found at ( 11 : 6 )
DEBUG LEXER —> T_CHAR [ s ] Found at ( 11 : 7 )
DEBUG LEXER —> T_CHAR [ t ] Found at ( 11 : 8 )
DEBUG LEXER —> T_CHAR [ r ] Found at ( 11 : 9 )
Error: (11:9) Unrecognized Token: 1
DEBUG LEXER —> T_CHAR [ n ] Found at ( 11 : 10 )
DEBUG LEXER —> T_CHAR [ g ] Found at ( 11 : 12 )
DEBUG LEXER —> T_QUOTE [ " ] Found at ( 11 : 13 )
DEBUG LEXER —> L_BRACE [ } ] Found at ( 12 : 1 )
DEBUG LEXER —> T_EOP [ $ ] Found at ( 12 : 2 )
INFO LEXER —> Lex completed with 1 errors
INFO LEXER —> Lex completed with 0 warnings
```

4.3 COMMENTS

This program results in an error because there is a digit located in a string and our grammar only allows chars, spaces, or nothing.

5 Test Case 5

5.1 Program

```
{
int a
a = a
string b
a = b
}$
```

5.2 output

```
INFO LEXER —> Lexing Program 5
DEBUG LEXER —> R_BRACE [ { ] Found at ( 14 : 1 )
DEBUG LEXER —> T_INT [ int ] Found at ( 15 : 3 )
DEBUG LEXER —> T_ID [ a ] Found at ( 15 : 7 )
DEBUG LEXER —> T_ID [ a ] Found at ( 16 : 3 )
DEBUG LEXER —> T_ASSIGN [ = ] Found at ( 16 : 5 )
DEBUG LEXER —> T_ID [ a ] Found at ( 16 : 7 )
DEBUG LEXER —> T_STRING [ string ] Found at ( 17 : 3 )
DEBUG LEXER —> T_ID [ b ] Found at ( 17 : 10 )
DEBUG LEXER —> T_ID [ a ] Found at ( 18 : 3 )
DEBUG LEXER —> T_ASSIGN [ = ] Found at ( 18 : 5 )
DEBUG LEXER —> T_ID [ b ] Found at ( 18 : 7 )
DEBUG LEXER —> L_BRACE [ } ] Found at ( 19 : 1 )
DEBUG LEXER —> T_EOP [ $ ] Found at ( 19 : 2 )
INFO LEXER —> Lex completed with 0 errors
INFO LEXER —> Lex completed with 0 warnings
```

5.3 comments

This program tests basic int and string declarations and completes with no errors or warnings.

6 Test Case 6

6.1 Program

```
{"inta"}$
```

6.2 output

```
INFO LEXER —> Lexing Program 6
DEBUG LEXER —> R_BRACE [ { ] Found at ( 21 : 1 )
DEBUG LEXER —> T_QUOTE [ " ] Found at ( 21 : 3 )
DEBUG LEXER —> T_CHAR [ i ] Found at ( 21 : 3 )
DEBUG LEXER —> T_CHAR [ n ] Found at ( 21 : 4 )
DEBUG LEXER —> T_CHAR [ t ] Found at ( 21 : 5 )
DEBUG LEXER —> T_CHAR [ a ] Found at ( 21 : 6 )
```

DEBUG LEXER —> T_QUOTE ["] Found at (21 : 7)
DEBUG LEXER —> L_BRACE [}] Found at (21 : 10)
DEBUG LEXER —> T_EOP [$] Found at (21 : 11)
INFO LEXER —> Lex completed with 0 errors
INFO LEXER —> Lex completed with 0 warnings

6.3 COMMENTS

This shows that the keyword int is not recognized if inside a string.

7 TEST CASE 7

7.1 PROGRAM

/*LongTestCase-EverythingExceptBooleanDeclaration*/{/*IntDeclaration*/intaintba=0b=0/*WhileLoop*/while(a!=3){pri
is no spoon"/*Thiswilldonothing*/)}}b=0a=1+a}}$

7.2 OUTPUT

INFO LEXER —> Lexing Program 7
DEBUG LEXER —> R_BRACE [{] Found at (23 : 52)
DEBUG LEXER —> T_INT [int] Found at (23 : 70)
DEBUG LEXER —> T_ID [a] Found at (23 : 72)
DEBUG LEXER —> T_INT [int] Found at (23 : 74)
DEBUG LEXER —> T_ID [b] Found at (23 : 76)
DEBUG LEXER —> T_ID [a] Found at (23 : 77)
DEBUG LEXER —> T_ASSIGN [=] Found at (23 : 80)
DEBUG LEXER —> T_DIGIT [0] Found at (23 : 81)
DEBUG LEXER —> T_ID [b] Found at (23 : 80)
DEBUG LEXER —> T_ASSIGN [=] Found at (23 : 83)
DEBUG LEXER —> T_DIGIT [0] Found at (23 : 83)
DEBUG LEXER —> T_WHILE [while] Found at (23 : 97)
DEBUG LEXER —> R_PAREN [( ] Found at (23 : 101)
DEBUG LEXER —> T_ID [a] Found at (23 : 103)
DEBUG LEXER —> T_UNEQUAL [!=] Found at (23 : 106)
DEBUG LEXER —> T_DIGIT [3] Found at (23 : 107)
DEBUG LEXER —> L_PAREN [ )] Found at (23 : 108)
DEBUG LEXER —> R_BRACE [{] Found at (23 : 109)
DEBUG LEXER —> T_PRINT [print] Found at (23 : 109)
DEBUG LEXER —> R_PAREN [( ] Found at (23 : 113)
DEBUG LEXER —> T_ID [a] Found at (23 : 114)
DEBUG LEXER —> L_PAREN [ )] Found at (23 : 117)
DEBUG LEXER —> T_WHILE [while] Found at (23 : 117)
DEBUG LEXER —> R_PAREN [( ] Found at (23 : 121)
DEBUG LEXER —> T_ID [b] Found at (23 : 123)
DEBUG LEXER —> T_UNEQUAL [!=] Found at (23 : 126)
DEBUG LEXER —> T_DIGIT [3] Found at (23 : 127)
DEBUG LEXER —> L_PAREN [ )] Found at (23 : 128)
DEBUG LEXER —> R_BRACE [{] Found at (23 : 129)

DEBUG LEXER —> T_PRINT [print] Found at (23 : 129)
DEBUG LEXER —> R_PAREN [( ] Found at (23 : 133)
DEBUG LEXER —> T_ID [b] Found at (23 : 134)
DEBUG LEXER —> L_PAREN [ )] Found at (23 : 137)
DEBUG LEXER —> T_ID [b] Found at (23 : 136)
DEBUG LEXER —> T_ASSIGN [=] Found at (23 : 139)
DEBUG LEXER —> T_DIGIT [1] Found at (23 : 140)
DEBUG LEXER —> T_INTOP [+] Found at (23 : 141)
DEBUG LEXER —> T_ID [b] Found at (23 : 142)
DEBUG LEXER —> T_IF [if] Found at (23 : 142)
DEBUG LEXER —> R_PAREN [( ] Found at (23 : 143)
DEBUG LEXER —> T_ID [b] Found at (23 : 145)
DEBUG LEXER —> T_EQUAL [==] Found at (23 : 148)
DEBUG LEXER —> T_DIGIT [2] Found at (23 : 149)
DEBUG LEXER —> L_PAREN [ )] Found at (23 : 150)
DEBUG LEXER —> R_BRACE [{] Found at (23 : 151)
DEBUG LEXER —> T_PRINT [print] Found at (23 : 169)
DEBUG LEXER —> R_PAREN [( ] Found at (23 : 173)
DEBUG LEXER —> T_QUOTE ["] Found at (23 : 176)
DEBUG LEXER —> T_CHAR [t] Found at (23 : 177)
DEBUG LEXER —> T_CHAR [h] Found at (23 : 178)
DEBUG LEXER —> T_CHAR [e] Found at (23 : 179)
DEBUG LEXER —> T_CHAR [r] Found at (23 : 180)
DEBUG LEXER —> T_CHAR [e] Found at (23 : 181)
DEBUG LEXER —> T_CHAR [ ] Found at (23 : 182)
DEBUG LEXER —> T_CHAR [i] Found at (23 : 183)
DEBUG LEXER —> T_CHAR [s] Found at (23 : 184)
DEBUG LEXER —> T_CHAR [ ] Found at (23 : 185)
DEBUG LEXER —> T_CHAR [n] Found at (23 : 186)
DEBUG LEXER —> T_CHAR [o] Found at (23 : 187)
DEBUG LEXER —> T_CHAR [ ] Found at (23 : 188)
DEBUG LEXER —> T_CHAR [s] Found at (23 : 189)
DEBUG LEXER —> T_CHAR [p] Found at (23 : 190)
DEBUG LEXER —> T_CHAR [o] Found at (23 : 191)
DEBUG LEXER —> T_CHAR [o] Found at (23 : 192)
DEBUG LEXER —> T_CHAR [n] Found at (23 : 193)
DEBUG LEXER —> T_QUOTE ["] Found at (23 : 194)
DEBUG LEXER —> L_PAREN [ )] Found at (23 : 216)
DEBUG LEXER —> L_BRACE [}] Found at (23 : 217)
DEBUG LEXER —> L_BRACE [}] Found at (23 : 218)
DEBUG LEXER —> T_ID [b] Found at (23 : 217)
DEBUG LEXER —> T_ASSIGN [=] Found at (23 : 220)
DEBUG LEXER —> T_DIGIT [0] Found at (23 : 221)
DEBUG LEXER —> T_ID [a] Found at (23 : 220)
DEBUG LEXER —> T_ASSIGN [=] Found at (23 : 223)
DEBUG LEXER —> T_DIGIT [1] Found at (23 : 224)
DEBUG LEXER —> T_INTOP [+] Found at (23 : 225)
DEBUG LEXER —> T_ID [a] Found at (23 : 224)
DEBUG LEXER —> L_BRACE [}] Found at (23 : 227)
DEBUG LEXER —> L_BRACE [}] Found at (23 : 228)
DEBUG LEXER —> T_EOP [$] Found at (23 : 229)
INFO LEXER —> Lex completed with 0 errors

INFO LEXER —> Lex completed with 0 warnings

7.3 COMMENTS

This is the lex without spaces example

8 Test Case 8

8.1 Program

```
{
/* Int Declaration */
int a
int b
a = 0
b=0
/* While Loop */
while (a != 3) {
print(a)
while (b != 3) {
print(b)
b = 1 + b
if (b == 2) {
/* Print Statement */
print("there is no spoon" /* This will do nothing */ )
}
}
b = 0
a = 1+a
}
}$
```

8.2 OUTPUT

```
INFO LEXER —> Lexing Program 8
DEBUG LEXER —> R_BRACE [ { ] Found at ( 25 : 1 )
DEBUG LEXER —> T_INT [ int ] Found at ( 27 : 3 )
DEBUG LEXER —> T_ID [ a ] Found at ( 27 : 7 )
DEBUG LEXER —> T_INT [ int ] Found at ( 28 : 3 )
DEBUG LEXER —> T_ID [ b ] Found at ( 28 : 7 )
DEBUG LEXER —> T_ID [ a ] Found at ( 29 : 3 )
DEBUG LEXER —> T_ASSIGN [ = ] Found at ( 29 : 5 )
DEBUG LEXER —> T_DIGIT [ 0 ] Found at ( 29 : 7 )
DEBUG LEXER —> T_ID [ b ] Found at ( 30 : 1 )
DEBUG LEXER —> T_ASSIGN [ = ] Found at ( 30 : 4 )
DEBUG LEXER —> T_DIGIT [ 0 ] Found at ( 30 : 5 )
DEBUG LEXER —> T_WHILE [ while ] Found at ( 32 : 3 )
DEBUG LEXER —> R_PAREN [ ( ] Found at ( 32 : 9 )
DEBUG LEXER —> T_ID [ a ] Found at ( 32 : 10 )
```

DEBUG LEXER —> T_UNEQUAL [!=] Found at (32 : 12)
DEBUG LEXER —> T_DIGIT [3] Found at (32 : 15)
DEBUG LEXER —> L_PAREN [ )] Found at (32 : 16)
DEBUG LEXER —> R_BRACE [{] Found at (32 : 18)
DEBUG LEXER —> T_PRINT [print] Found at (33 : 7)
DEBUG LEXER —> R_PAREN [( ] Found at (33 : 11)
DEBUG LEXER —> T_ID [a] Found at (33 : 11)
DEBUG LEXER —> L_PAREN [ )] Found at (33 : 14)
DEBUG LEXER —> T_WHILE [while] Found at (34 : 7)
DEBUG LEXER —> R_PAREN [( ] Found at (34 : 13)
DEBUG LEXER —> T_ID [b] Found at (34 : 14)
DEBUG LEXER —> T_UNEQUAL [!=] Found at (34 : 16)
DEBUG LEXER —> T_DIGIT [3] Found at (34 : 19)
DEBUG LEXER —> L_PAREN [ )] Found at (34 : 20)
DEBUG LEXER —> R_BRACE [{] Found at (34 : 22)
DEBUG LEXER —> T_PRINT [print] Found at (35 : 11)
DEBUG LEXER —> R_PAREN [( ] Found at (35 : 15)
DEBUG LEXER —> T_ID [b] Found at (35 : 15)
DEBUG LEXER —> L_PAREN [ )] Found at (35 : 18)
DEBUG LEXER —> T_ID [b] Found at (36 : 11)
DEBUG LEXER —> T_ASSIGN [=] Found at (36 : 13)
DEBUG LEXER —> T_DIGIT [1] Found at (36 : 15)
DEBUG LEXER —> T_INTOP [+] Found at (36 : 18)
DEBUG LEXER —> T_ID [b] Found at (36 : 19)
DEBUG LEXER —> T_IF [if] Found at (37 : 11)
DEBUG LEXER —> R_PAREN [( ] Found at (37 : 14)
DEBUG LEXER —> T_ID [b] Found at (37 : 15)
DEBUG LEXER —> T_EQUAL [==] Found at (37 : 17)
DEBUG LEXER —> T_DIGIT [2] Found at (37 : 20)
DEBUG LEXER —> L_PAREN [ )] Found at (37 : 21)
DEBUG LEXER —> R_BRACE [{] Found at (37 : 23)
DEBUG LEXER —> T_PRINT [print] Found at (39 : 11)
DEBUG LEXER —> R_PAREN [( ] Found at (39 : 15)
DEBUG LEXER —> T_QUOTE ["] Found at (39 : 17)
DEBUG LEXER —> T_CHAR [t] Found at (39 : 18)
DEBUG LEXER —> T_CHAR [h] Found at (39 : 19)
DEBUG LEXER —> T_CHAR [e] Found at (39 : 20)
DEBUG LEXER —> T_CHAR [r] Found at (39 : 21)
DEBUG LEXER —> T_CHAR [e] Found at (39 : 22)
DEBUG LEXER —> T_CHAR [ ] Found at (39 : 23)
DEBUG LEXER —> T_CHAR [i] Found at (39 : 24)
DEBUG LEXER —> T_CHAR [s] Found at (39 : 25)
DEBUG LEXER —> T_CHAR [ ] Found at (39 : 26)
DEBUG LEXER —> T_CHAR [n] Found at (39 : 27)
DEBUG LEXER —> T_CHAR [o] Found at (39 : 28)
DEBUG LEXER —> T_CHAR [ ] Found at (39 : 29)
DEBUG LEXER —> T_CHAR [s] Found at (39 : 30)
DEBUG LEXER —> T_CHAR [p] Found at (39 : 31)
DEBUG LEXER —> T_CHAR [o] Found at (39 : 32)
DEBUG LEXER —> T_CHAR [o] Found at (39 : 33)
DEBUG LEXER —> T_CHAR [n] Found at (39 : 34)
DEBUG LEXER —> T_QUOTE ["] Found at (39 : 35)

DEBUG LEXER —> L_PAREN [ )] Found at (39 : 64)
DEBUG LEXER —> L_BRACE [}] Found at (39 : 12)
DEBUG LEXER —> L_BRACE [}] Found at (40 : 8)
DEBUG LEXER —> T_ID [b] Found at (42 : 7)
DEBUG LEXER —> T_ASSIGN [=] Found at (42 : 9)
DEBUG LEXER —> T_DIGIT [0] Found at (42 : 11)
DEBUG LEXER —> T_ID [a] Found at (43 : 7)
DEBUG LEXER —> T_ASSIGN [=] Found at (43 : 9)
DEBUG LEXER —> T_DIGIT [1] Found at (43 : 11)
DEBUG LEXER —> T_INTOP [+] Found at (43 : 12)
DEBUG LEXER —> T_ID [a] Found at (43 : 13)
DEBUG LEXER —> L_BRACE [}] Found at (43 : 4)
DEBUG LEXER —> L_BRACE [}] Found at (45 : 1)
DEBUG LEXER —> T_EOP [$] Found at (45 : 2)
INFO LEXER —> Lex completed with 0 errors
INFO LEXER —> Lex completed with 0 warnings

8.3 COMMENTS

This program is the same as the one above, hence the output is the same regardless of the whitespace boundaries.

9 Test Case 9

9.1 Program

{/* comments are still ignored */ int@}$

9.2 OUTPUT

INFO LEXER —> Lexing Program 9
DEBUG LEXER —> R_BRACE [{] Found at (47 : 1)
Error: (47:38) Unrecognized Token: @
DEBUG LEXER —> T_INT [int] Found at (47 : 34)
DEBUG LEXER —> [@] Found at (47 : 36)
DEBUG LEXER —> L_BRACE [}] Found at (47 : 39)
DEBUG LEXER —> T_EOP [$] Found at (47 : 40)
INFO LEXER —> Lex completed with 1 errors
INFO LEXER —> Lex completed with 0 warnings

9.3 COMMENTS

The @ token is not apart of the grammar, therefore an error is thrown

10 Test Case 10

10.1 Program

{/*bad comment }

10.2 output

INFO LEXER —> Lexing Program 10
DEBUG LEXER —> R_BRACE [{] Found at (49 : 1)
Warning: Comment did not close
INFO LEXER —> Lex completed with 0 errors
INFO LEXER —> Lex completed with 1 warnings

10.3 comments

This program shows the warning thrown if a comment is not closed.

11 Test Case 11

11.1 Program

{}

11.2 output

INFO LEXER —> Lexing Program 11
DEBUG LEXER —> R_BRACE [{] Found at (51 : 1)
DEBUG LEXER —> L_BRACE [}] Found at (51 : 2)
Warning: Missing End of Program char: $

11.3 comments

The final test case shows the warning thrown if there is no end of program symbol found.