Lexical Analysis Testing Results

Emily Doran

Emily.Doran1@Marist.edu

March 9, 2021

Basic Test Cases (Verbose Mode)

Input text:

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

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

Output Token Sequence:

```
INFO  Lexer - Lexing program 1...
DEBUG Lexer - T_L_BRACE [ { ] found at (1:1)
DEBUG Lexer - T_R_BRACE [ } ] found at (1:2)
DEBUG Lexer - T_EOP [ $ ] found at (1:3)
INFO  Lexer - Lex completed with 0 errors

INFO  Lexer - Lexing program 2...
DEBUG Lexer - T_L_BRACE [ { ] found at (2:1)
DEBUG Lexer - T_L_BRACE [ { ] found at (2:2)
DEBUG Lexer - T_L_BRACE [ { ] found at (2:3)
DEBUG Lexer - T_L_BRACE [ { ] found at (2:4)
DEBUG Lexer - T_L_BRACE [ { ] found at (2:5)
DEBUG Lexer - T_L_BRACE [ { ] found at (2:6)
DEBUG Lexer - T_R_BRACE [ } ] found at (2:7)
DEBUG Lexer - T_R_BRACE [ } ] found at (2:8)
```

```
DEBUG Lexer - T_R_BRACE [ } ] found at (2:9)
DEBUG Lexer - T_R_BRACE [ } ] found at (2:10)
DEBUG Lexer - T_R_BRACE [ } ] found at (2:11)
DEBUG Lexer - T_R_BRACE [ } ] found at (2:12)
DEBUG Lexer - T_EOP [ $ ] found at (2:13)
INFO  Lexer - Lex completed with 0 errors

INFO  Lexer - Lexing program 3...
DEBUG Lexer - T_L_BRACE [ { ] found at (3:1)
DEBUG Lexer - T_L_BRACE [ { ] found at (3:2)
DEBUG Lexer - T_L_BRACE [ { ] found at (3:3)
DEBUG Lexer - T_L_BRACE [ { ] found at (3:4)
DEBUG Lexer - T_L_BRACE [ { ] found at (3:5)
DEBUG Lexer - T_L_BRACE [ { ] found at (3:6)
DEBUG Lexer - T_R_BRACE [ } ] found at (3:7)
DEBUG Lexer - T_R_BRACE [ } ] found at (3:8)
DEBUG Lexer - T_R_BRACE [ } ] found at (3:9)
DEBUG Lexer - T_R_BRACE [ } ] found at (3:38)
DEBUG Lexer - T_R_BRACE [ } ] found at (3:39)
DEBUG Lexer - T_R_BRACE [ } ] found at (3:40)
DEBUG Lexer - T_R_BRACE [ } ] found at (3:41)
DEBUG Lexer - T_EOP [ $ ] found at (3:42)
INFO  Lexer - Lex completed with 0 errors

INFO  Lexer - Lexing program 4...
DEBUG Lexer - T_L_BRACE [ { ] found at (4:1)
DEBUG Lexer - T_VARIABLE_TYPE [ int ] found at (4:36)
ERROR Lexer - Error: 4:40 Unrecognized Token: @
DEBUG Lexer - T_R_BRACE [ } ] found at (4:41)
DEBUG Lexer - T_EOP [ $ ] found at (4:42)
ERROR Lexer - Lex failed with 1 error(s)

INFO  Lexer - Lexing program 5...
DEBUG Lexer - T_L_BRACE [ { ] found at (6:1)
DEBUG Lexer - T_VARIABLE_TYPE [ int ] found at (7:3)
DEBUG Lexer - T_ID [ a ] found at (7:7)
DEBUG Lexer - T_ID [ a ] found at (8:3)
DEBUG Lexer - T_ASSIGN_OP [ = ] found at (8:5)
DEBUG Lexer - T_ID [ a ] found at (8:7)
DEBUG Lexer - T_VARIABLE_TYPE [ string ] found at (9:3)
DEBUG Lexer - T_ID [ b ] found at (9:10)
DEBUG Lexer - T_ID [ a ] found at (10:3)
DEBUG Lexer - T_ASSIGN_OP [ = ] found at (10:5)
DEBUG Lexer - T_ID [ b ] found at (10:7)
DEBUG Lexer - T_R_BRACE [ } ] found at (11:1)
DEBUG Lexer - T_EOP [ $ ] found at (11:2)
INFO  Lexer - Lex completed with 0 errors
```

All programs complete the lexical analysis without any errors except for program 4. Program 4 fails with an error because an invalid character, '@', was entered. Since this character isn't in our grammar, entering it anywhere except within a comment will throw an error.

Lex Without Spaces (Verbose Mode)

Input text:

```
{intaintba=0b=0while(a!=3) {print(a)while(b!=3) {print(b)b=1+bif(b==2){print("there
is no spoon")}}b=0a=1+a}}$
```

Output Token Sequence:

```
INFO  Lexer - Lexing program 1...
DEBUG Lexer - T_L_BRACE [ { ] found at (1:1)
DEBUG Lexer - T_VARIABLE_TYPE [ int ] found at (1:2)
DEBUG Lexer - T_ID [ a ] found at (1:5)
DEBUG Lexer - T_VARIABLE_TYPE [ int ] found at (1:6)
DEBUG Lexer - T_ID [ b ] found at (1:9)
DEBUG Lexer - T_ID [ a ] found at (1:10)
DEBUG Lexer - T_ASSIGN_OP [ = ] found at (1:11)
DEBUG Lexer - T_DIGIT [ 0 ] found at (1:12)
DEBUG Lexer - T_ID [ b ] found at (1:13)
DEBUG Lexer - T_ASSIGN_OP [ = ] found at (1:14)
DEBUG Lexer - T_DIGIT [ 0 ] found at (1:15)
DEBUG Lexer - T_WHILE [ while ] found at (1:16)
DEBUG Lexer - T_L_PAREN [ ( ] found at (1:21)
DEBUG Lexer - T_ID [ a ] found at (1:22)
DEBUG Lexer - T_INEQUALITY_OP [ != ] found at (1:23)
DEBUG Lexer - T_DIGIT [ 3 ] found at (1:25)
DEBUG Lexer - T_R_PAREN [ ) ] found at (1:26)
DEBUG Lexer - T_L_BRACE [ { ] found at (1:28)
DEBUG Lexer - T_PRINT] [ print ] found at (1:29)
DEBUG Lexer - T_L_PAREN [ ( ] found at (1:34)
DEBUG Lexer - T_ID [ a ] found at (1:35)
DEBUG Lexer - T_R_PAREN [ ) ] found at (1:36)
DEBUG Lexer - T_WHILE [ while ] found at (1:37)
DEBUG Lexer - T_L_PAREN [ ( ] found at (1:42)
DEBUG Lexer - T_ID [ b ] found at (1:43)
DEBUG Lexer - T_INEQUALITY_OP [ != ] found at (1:44)
DEBUG Lexer - T_DIGIT [ 3 ] found at (1:46)
DEBUG Lexer - T_R_PAREN [ ) ] found at (1:47)
DEBUG Lexer - T_L_BRACE [ { ] found at (1:49)
DEBUG Lexer - T_PRINT] [ print ] found at (1:50)
DEBUG Lexer - T_L_PAREN [ ( ] found at (1:55)
DEBUG Lexer - T_ID [ b ] found at (1:56)
DEBUG Lexer - T_R_PAREN [ ) ] found at (1:57)
DEBUG Lexer - T_ID [ b ] found at (1:58)
DEBUG Lexer - T_ASSIGN_OP [ = ] found at (1:59)
DEBUG Lexer - T_DIGIT [ 1 ] found at (1:60)
DEBUG Lexer - T_ADDITION_OP [ + ] found at (1:61)
DEBUG Lexer - T_ID [ b ] found at (1:62)
DEBUG Lexer - T_IF] [ if ] found at (1:63)
DEBUG Lexer - T_L_PAREN [ ( ] found at (1:65)
DEBUG Lexer - T_ID [ b ] found at (1:66)
DEBUG Lexer - T_EQUALITY_OP [ == ] found at (1:67)
DEBUG Lexer - T_DIGIT [ 2 ] found at (1:69)
```

```
DEBUG Lexer - T_R_PAREN [ ) ] found at (1:70)
DEBUG Lexer - T_L_BRACE [ { ] found at (1:71)
DEBUG Lexer - T_PRINT] [ print ] found at (1:72)
DEBUG Lexer - T_L_PAREN [ ( ] found at (1:77)
DEBUG Lexer - T_QUOTE [ " ] found at (1:78)
DEBUG Lexer - T_CHAR [ t ] found at (1:79)
DEBUG Lexer - T_CHAR [ h ] found at (1:80)
DEBUG Lexer - T_CHAR [ e ] found at (1:81)
DEBUG Lexer - T_CHAR [ r ] found at (1:82)
DEBUG Lexer - T_CHAR [ e ] found at (1:83)
DEBUG Lexer - T_CHAR [   ] found at (1:84)
DEBUG Lexer - T_CHAR [ i ] found at (1:85)
DEBUG Lexer - T_CHAR [ s ] found at (1:86)
DEBUG Lexer - T_CHAR [   ] found at (1:87)
DEBUG Lexer - T_CHAR [ n ] found at (1:88)
DEBUG Lexer - T_CHAR [ o ] found at (1:89)
DEBUG Lexer - T_CHAR [   ] found at (1:90)
DEBUG Lexer - T_CHAR [ s ] found at (1:91)
DEBUG Lexer - T_CHAR [ p ] found at (1:92)
DEBUG Lexer - T_CHAR [ o ] found at (1:93)
DEBUG Lexer - T_CHAR [ o ] found at (1:94)
DEBUG Lexer - T_CHAR [ n ] found at (1:95)
DEBUG Lexer - T_QUOTE [ " ] found at (1:96)
DEBUG Lexer - T_R_PAREN [ ) ] found at (1:97)
DEBUG Lexer - T_R_BRACE [ } ] found at (1:98)
DEBUG Lexer - T_R_BRACE [ } ] found at (1:99)
DEBUG Lexer - T_ID [ b ] found at (1:100)
DEBUG Lexer - T_ASSIGN_OP [ = ] found at (1:101)
DEBUG Lexer - T_DIGIT [ 0 ] found at (1:102)
DEBUG Lexer - T_ID [ a ] found at (1:103)
DEBUG Lexer - T_ASSIGN_OP [ = ] found at (1:104)
DEBUG Lexer - T_DIGIT [ 1 ] found at (1:105)
DEBUG Lexer - T_ADDITION_OP [ + ] found at (1:106)
DEBUG Lexer - T_ID [ a ] found at (1:107)
DEBUG Lexer - T_R_BRACE [ } ] found at (1:108)
DEBUG Lexer - T_R_BRACE [ } ] found at (1:109)
DEBUG Lexer - T_EOP [ $ ] found at (1:110)
INFO  Lexer - Lex completed with 0 errors
```

Although there are no spaces or new lines in this program, it still completes the lexical analysis without any errors. All of the characters that were entered were valid in our grammar.

Throw Warning - Unclosed String (Verbose Mode)

Input text:

```
{"this is an unclosed string}$
```

Output Token Sequence:

```
INFO  Lexer - Lexing program 1...
DEBUG Lexer - T_L_BRACE [ { ] found at (1:1)
```

```
DEBUG Lexer - T_QUOTE [ " ] found at (1:2)
DEBUG Lexer - T_CHAR [ t ] found at (1:3)
DEBUG Lexer - T_CHAR [ h ] found at (1:4)
DEBUG Lexer - T_CHAR [ i ] found at (1:5)
DEBUG Lexer - T_CHAR [ s ] found at (1:6)
DEBUG Lexer - T_CHAR [   ] found at (1:7)
DEBUG Lexer - T_CHAR [ i ] found at (1:8)
DEBUG Lexer - T_CHAR [ s ] found at (1:9)
DEBUG Lexer - T_CHAR [   ] found at (1:10)
DEBUG Lexer - T_CHAR [ a ] found at (1:11)
DEBUG Lexer - T_CHAR [ n ] found at (1:12)
DEBUG Lexer - T_CHAR [   ] found at (1:13)
DEBUG Lexer - T_CHAR [ u ] found at (1:14)
DEBUG Lexer - T_CHAR [ n ] found at (1:15)
DEBUG Lexer - T_CHAR [ c ] found at (1:16)
DEBUG Lexer - T_CHAR [ l ] found at (1:17)
DEBUG Lexer - T_CHAR [ o ] found at (1:18)
DEBUG Lexer - T_CHAR [ s ] found at (1:19)
DEBUG Lexer - T_CHAR [ e ] found at (1:20)
DEBUG Lexer - T_CHAR [ d ] found at (1:21)
DEBUG Lexer - T_CHAR [   ] found at (1:22)
DEBUG Lexer - T_CHAR [ s ] found at (1:23)
DEBUG Lexer - T_CHAR [ t ] found at (1:24)
DEBUG Lexer - T_CHAR [ r ] found at (1:25)
DEBUG Lexer - T_CHAR [ i ] found at (1:26)
DEBUG Lexer - T_CHAR [ n ] found at (1:27)
DEBUG Lexer - T_CHAR [ g ] found at (1:28)
ERROR Lexer - Error: 1:29 Unrecognized Token inside string: }
ERROR Lexer - Error: 1:30 Unrecognized Token inside string: $
WARNING Lexer - Missing EOP Character '$'
WARNING Lexer - Unclosed String at End of Program
```

Since we do not close the string before the end of program is reached, errors are thrown for the right brace and EOP symbols because they are considered to be within the string still. Warnings are then thrown to let the user know there is a string that was never closed and that they are missing the EOP character.

Throw Warning - Unclosed Comment (Verbose Mode)

Input text:

```
{/* this is an unclosed comment }$
```

Output Token Sequence:

```
INFO  Lexer - Lexing program 1...
DEBUG Lexer - T_L_BRACE [ { ] found at (1:1)
WARNING Lexer - Missing EOP Character '$'
WARNING Lexer - Unclosed Comment at End of Program
```

This program correctly finds the left brace, then finds a comment, so it ignores everything within the comment. Since the comment is never closed, a warning is output letting the user know they never closed their comment and that they are missing the EOP character since it was never found since we are still technically inside a comment.

Throw Warning - Missing EOP (Verbose Mode)

Input text:

```
{int a=2 print("i am missing the eop char")}
```

Output Token Sequence:

```
INFO  Lexer - Lexing program 1...
DEBUG Lexer - T_L_BRACE [ { ] found at (1:1)
DEBUG Lexer - T_VARIABLE_TYPE [ int ] found at (1:2)
DEBUG Lexer - T_ID [ a ] found at (1:6)
DEBUG Lexer - T_ASSIGN_OP [ = ] found at (1:7)
DEBUG Lexer - T_DIGIT [ 2 ] found at (1:8)
DEBUG Lexer - T_PRINT] [ print ] found at (1:10)
DEBUG Lexer - T_L_PAREN [ ( ] found at (1:15)
DEBUG Lexer - T_QUOTE [ " ] found at (1:16)
DEBUG Lexer - T_CHAR [ i ] found at (1:17)
DEBUG Lexer - T_CHAR [   ] found at (1:18)
DEBUG Lexer - T_CHAR [ a ] found at (1:19)
DEBUG Lexer - T_CHAR [ m ] found at (1:20)
DEBUG Lexer - T_CHAR [   ] found at (1:21)
DEBUG Lexer - T_CHAR [ m ] found at (1:22)
DEBUG Lexer - T_CHAR [ i ] found at (1:23)
DEBUG Lexer - T_CHAR [ s ] found at (1:24)
DEBUG Lexer - T_CHAR [ s ] found at (1:25)
DEBUG Lexer - T_CHAR [ i ] found at (1:26)
DEBUG Lexer - T_CHAR [ n ] found at (1:27)
DEBUG Lexer - T_CHAR [ g ] found at (1:28)
DEBUG Lexer - T_CHAR [   ] found at (1:29)
DEBUG Lexer - T_CHAR [ t ] found at (1:30)
DEBUG Lexer - T_CHAR [ h ] found at (1:31)
DEBUG Lexer - T_CHAR [ e ] found at (1:32)
DEBUG Lexer - T_CHAR [   ] found at (1:33)
DEBUG Lexer - T_CHAR [ e ] found at (1:34)
DEBUG Lexer - T_CHAR [ o ] found at (1:35)
DEBUG Lexer - T_CHAR [ p ] found at (1:36)
DEBUG Lexer - T_CHAR [   ] found at (1:37)
DEBUG Lexer - T_CHAR [ c ] found at (1:38)
DEBUG Lexer - T_CHAR [ h ] found at (1:39)
DEBUG Lexer - T_CHAR [ a ] found at (1:40)
DEBUG Lexer - T_CHAR [ r ] found at (1:41)
DEBUG Lexer - T_QUOTE [ " ] found at (1:42)
DEBUG Lexer - T_R_PAREN [ ) ] found at (1:43)
DEBUG Lexer - T_R_BRACE [ } ] found at (1:44)
WARNING Lexer - Missing EOP Character '$'
```

All necessary tokens are recognized in this program. Since we are missing the EOP character and have reached the end of the file, a warning is output. Since we aren't inside a string or comment still, the EOP character is added to the end so that the program will continue for later project parts without errors.

Throw Error - Invalid Tokens in String (Verbose Mode)

Input text:

```
{if(x!=1)b="symbols throw errors!!"}$
{"digits throw errors2"}$
```

Output Token Sequence:

```
INFO  Lexer - Lexing program 1...
DEBUG Lexer - T_L_BRACE [ { ] found at (1:1)
DEBUG Lexer - T_IF] [ if ] found at (1:2)
DEBUG Lexer - T_L_PAREN [ ( ] found at (1:4)
DEBUG Lexer - T_ID [ x ] found at (1:5)
DEBUG Lexer - T_INEQUALITY_OP [ != ] found at (1:6)
DEBUG Lexer - T_DIGIT [ 1 ] found at (1:8)
DEBUG Lexer - T_R_PAREN [ ) ] found at (1:9)
DEBUG Lexer - T_ID [ b ] found at (1:10)
DEBUG Lexer - T_ASSIGN_OP [ = ] found at (1:11)
DEBUG Lexer - T_QUOTE [ " ] found at (1:12)
DEBUG Lexer - T_CHAR [ s ] found at (1:13)
DEBUG Lexer - T_CHAR [ y ] found at (1:14)
DEBUG Lexer - T_CHAR [ m ] found at (1:15)
DEBUG Lexer - T_CHAR [ b ] found at (1:16)
DEBUG Lexer - T_CHAR [ o ] found at (1:17)
DEBUG Lexer - T_CHAR [ l ] found at (1:18)
DEBUG Lexer - T_CHAR [ s ] found at (1:19)
DEBUG Lexer - T_CHAR [   ] found at (1:20)
DEBUG Lexer - T_CHAR [ t ] found at (1:21)
DEBUG Lexer - T_CHAR [ h ] found at (1:22)
DEBUG Lexer - T_CHAR [ r ] found at (1:23)
DEBUG Lexer - T_CHAR [ o ] found at (1:24)
DEBUG Lexer - T_CHAR [ w ] found at (1:25)
DEBUG Lexer - T_CHAR [   ] found at (1:26)
DEBUG Lexer - T_CHAR [ e ] found at (1:27)
DEBUG Lexer - T_CHAR [ r ] found at (1:28)
DEBUG Lexer - T_CHAR [ r ] found at (1:29)
DEBUG Lexer - T_CHAR [ o ] found at (1:30)
DEBUG Lexer - T_CHAR [ r ] found at (1:31)
DEBUG Lexer - T_CHAR [ s ] found at (1:32)
ERROR Lexer - Error: 1:33 Unrecognized Token inside string: !
ERROR Lexer - Error: 1:34 Unrecognized Token inside string: !
DEBUG Lexer - T_QUOTE [ " ] found at (1:35)
DEBUG Lexer - T_R_BRACE [ } ] found at (1:36)
DEBUG Lexer - T_EOP [ $ ] found at (1:37)
ERROR Lexer - Lex failed with 2 error(s)

INFO  Lexer - Lexing program 2...
DEBUG Lexer - T_L_BRACE [ { ] found at (2:1)
DEBUG Lexer - T_QUOTE [ " ] found at (2:2)
DEBUG Lexer - T_CHAR [ d ] found at (2:3)
DEBUG Lexer - T_CHAR [ i ] found at (2:4)
DEBUG Lexer - T_CHAR [ g ] found at (2:5)
```

```
DEBUG Lexer - T_CHAR [ i ] found at (2:6)
DEBUG Lexer - T_CHAR [ t ] found at (2:7)
DEBUG Lexer - T_CHAR [ s ] found at (2:8)
DEBUG Lexer - T_CHAR [   ] found at (2:9)
DEBUG Lexer - T_CHAR [ t ] found at (2:10)
DEBUG Lexer - T_CHAR [ h ] found at (2:11)
DEBUG Lexer - T_CHAR [ r ] found at (2:12)
DEBUG Lexer - T_CHAR [ o ] found at (2:13)
DEBUG Lexer - T_CHAR [ w ] found at (2:14)
DEBUG Lexer - T_CHAR [   ] found at (2:15)
DEBUG Lexer - T_CHAR [ e ] found at (2:16)
DEBUG Lexer - T_CHAR [ r ] found at (2:17)
DEBUG Lexer - T_CHAR [ r ] found at (2:18)
DEBUG Lexer - T_CHAR [ o ] found at (2:19)
DEBUG Lexer - T_CHAR [ r ] found at (2:20)
DEBUG Lexer - T_CHAR [ s ] found at (2:21)
ERROR Lexer - Error: 2:22 Unrecognized Token inside string: 2
DEBUG Lexer - T_QUOTE [ " ] found at (2:23)
DEBUG Lexer - T_R_BRACE [ } ] found at (2:24)
DEBUG Lexer - T_EOP [ $ ] found at (2:25)
ERROR Lexer - Lex failed with 1 error(s)
```

The first program fails with 2 errors because of the '!!' at the end of the string. Strings in our language do not allow anything except nothing, spaces, or characters in strings. The second program fails for the same reason because a digit is entered in the string.

Test Cases Non Verbose Mode

Input text:

```
{intabooleanba=0b=false}$
{"hello" boolean a = true}$

/*Long Test Case - Everything Except Boolean Declaration */
{
  /* 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
  }
}$
```

Output:

```
INFO  Lexer - Lexing program 1...
INFO  Lexer - Lex completed with 0 errors

INFO  Lexer - Lexing program 2...
INFO  Lexer - Lex completed with 0 errors

INFO  Lexer - Lexing program 3...
INFO  Lexer - Lex completed with 0 errors
```

When running the lexical analyzer in non-verbose test mode, each individual DEBUG token is not printed out. Only messages when you begin lexing, any errors or warnings, and the complete/fail message are printed.

Test Cases Non Verbose Mode - Error

Input text:

```
{"this will throw an error in a string $"}$
```

Output:

```
INFO  Lexer - Lexing program 1...
ERROR Lexer - Error: 1:40 Unrecognized Token inside string: $
ERROR Lexer - Lex failed with 1 error(s)
```

Since we are running this in non-verbose test mode, only the starting INFO message, the error message, and the failed message are printed. This program failed because we entered an EOP symbol within a string which is not allowed.

Test Cases Non Verbose Mode - Warning

Input text:

```
{{{{{{}}}}}}
```

Output:

```
INFO  Lexer - Lexing program 1...
WARNING Lexer - Missing EOP Character '$'
```

Warning messages are still printed when running in non-verbose test mode. Since we are missing the EOP character and have reached the end of the file, a warning is output and the symbol is added to the end of the file.