

CMSC 430 Project 1

Author: Tyler D Clark

Date: 1 November 2021

Description The first project involved modifying a lexical analyzer and compilation listing generator code.

Approach

In preparation for this project, I needed to understand how flex operated. I read the required material, watched the videos and got to work. With the knowledge I gained, I was able to add in the reserved words and lexemes. Everything I needed to accomplish this project was provided through the reading and videos. The only thing I needed outside of the material was a brush up on regex and a review of the flex code.

I first added in the reserved words, because it was as easy as listing the words and returned the corresponding tokens. I then added in the lexemes that required regex. This was the part that required more work as I had to look up the regex syntax. Lastly, I added in the code for the listing generator.

Test cases

Test 1

The first test case was to test all of the lexemes and keywords in the language. The lexical analyzer was modified to recognize all of those keywords and lexemes listed in the directions.

input: test1.txt

output:

```
tylerclark@MacBook-Pro:~/Repos/CMSC430/Project1
λ ./compile < tests/test1.txt (main x!?)
1 // Second comment
2
3 -- Punctuation symbols
4
5 ,;() =>
6
7 -- Identifier
8
9 name name123 name_123 name_123_name
10
11 -- Literals
12
13 123 100.0 1.0e2 1.0e-2 true false
14
15 -- Logical operators
16
17 and or not
18
19 -- Relational operators
20
21 < = /= > ≥ ≤
22
23 -- Arithmetic operators
24
25 + - * ** / rem
26
27 -- Reserved words
28
29 begin else end endif endreduce function if is integer reduce returns then case endcase others real when
Compiled Successfully
tylerclark ~/.../CMSC430/Project1
λ (main x!?)
```

Test 2

The next test case was to test the lexical analyzers ability to detect and report multiple errors. This test case is borrowed from the week 2 resources.

input: test2.txt

output:

```
tylerclark ~/.../CMSC430/Project1
λ ./compile < tests/test2.txt (main x!?)

1 -- Function with two lexical errors
2
3 function test2 returns integer;
4 begin
5     7 $ 2 ^ (2 + 4);
Lexical Error, Invalid Character $
Lexical Error, Invalid Character ^
6 end;

Lexical Errors: 2
Syntax Errors: 0
Semantic Errors: 0
tylerclark ~/.../CMSC430/Project1
λ
```

Test 3

The last case is a program with no errors. This test case is also borrowed from the week 2 resources.

```
tylerclark ~/.../CMSC430/Project1
λ ./compile < tests/test3.txt (main x!?)

1 -- Program with no errors
2
3 function test3 returns boolean;
4 begin
5     7 + 2 > 6 and 8 = 5 * (7 - 4);
6 end;

Compiled Successfully
tylerclark ~/.../CMSC430/Project1
λ
```

Lessons Learned

Through this project, I gained valuable information about flex, regex, C++ and how to use them together. I also learned how to use the flex code to create a lexical analyzer. I did not experience any major hiccups, but I did spend a lot of time trying to make the regex work. This project was also a good brush up on C++ and I always enjoy using it. This is also the first time I have used make instead of CMake, so I learned as much as I could about the makefile and even included a clean command. Given the amount of time I spent on this project, I am confident that I will be able to build off of this information in future projects.