Dalton Stokke
Qin Xukun
Writing Assignment 1

Ways to Improve Iteration 1 Scanner Implementation

The purpose of this writing assignment is to identify and fix potential undesirable methods of implementation for our Scanner in Iteration 1. We are asked to specifically consider the following six potential problems:

1. Is there any duplication in the testing code you've written to test individual token regular expressions and the code used by scan?

2. Does your scan function make calls to makeRegex every time it is called?

3. Do you have redundant arrays of TokenType values?

4. Would changing the order of kVariableKwd and kEndKwd in the definition of enum kTokenEnumType in scanner.h (or your implementations relevant file) have any adverse effect on the rest of your code?

5. Do you create a name regex_t pointer for each regex instead of only putting them in an array?

6. Are there any places in which enumerated TokenType values should be used instead of integer literals?

After evaluation of our code we discovered that we do not properly handle the issues described in problem 2, 4, 5, and 6. Specifically, our Scanner 1 implementation creates new regexs each time Scan is called, and in doing so assigned a named regex_t pointer for each regex before putting them in an array, who's indexes we then referred to by integer literals instead of using the relevant enumerated TokenType. This effected problem 4, because our array indexing would not adapt to changes made to the enum TokenType. Overall, this meant that for every call to scan we instantiated multiple pointers on the stack, wasting cycles and memory resources on something that only needed to be instantiated once. And referring to our indexes of that array as integer literals reduced clarity in our code, while creating a dependency on the ordering of the enum TokenType. To fix these issues, we created a Scanner constructor and instantiated our regex_t array of all regexes by the index values of the relevant TokenType, so that upon creating Scanner object our array will be instantiated only once.

As for problems 1 and 3, our code does not exhibit these issues. If we had the issue of problem 1, it would mean an increase in redundancy in our code and wasted resources, including cpu cycles and memory. If we had the issue of problem 3 and we created an array, which isn't needed as we already have enum TokeyType that we can reference by its ordinal values, then we would waste memory on a redundant array.

In conclusion, we were able to fix all issues in our code by creating a Scanner constructor and instantiating our regex_t array of regexes using the index values of the relevant TokenType. Doing so was a very easy fix that saved memory, due to multiple pointers and regex_t arrays, and cpu cycles that would have been required to create them, resulting in a significant improvement, cutting down on over 40 pointers worth of stack space and severely reducing the number of calls to make_regex.