Compiler Lab2 Report

Haocheng Wang 521021910225

September 24, 2023

1 how to handle comments

I set a unique COMMENT state to handle text stream in the comments. Anyway, I use the comment_level_ data member in Scanner class to record the level of the comment being scanned.

The comment_level_ is initialized 0 when scanner was created. When I scan an /* in the text stream, we add 1 to comment_level_ , and transfer to COMMENT state.

In the COMMENT state, we will handle three tokens:

- /*: will make the comment_level_ increase 1 level
- */: will make the comment_level_ decrease 1 level. if the level is 0, then exit to INITIAL state. Finish a comment scan
- other char: ignore

2 how to handle strings

I set a unique STRING state to handle text stream in string.And like the way mentioned above, we use some utils in Scanner class to help complete the task. For example, we use the d_matched to get current matched string by call matched() and setMatched(), we use d_matched data member to record all the analyzed content in this string scan task.

When I scan " in the text stream in INITIAL state. We empty the string_buf_ , use adjustStr()(this function can make test scan cursor go ahead as normal but also record the start position of the string), and enter the STRING state.

In the STRING state,we will handle there tokens:

- 7 kinds of escape text:
  - \n
  - \t
  - \^C
  - \ddd
  - \"
  - \\
  - \f___f\

  I will match them, and append the real char to the string_buf_ string.
- ": I will then set the d_matched as the value of string_buf_ (the d_matched will be used to print in the test program), empty the string_buf_ , exit to the INITIAL state and return a Parser::STRING.
- other char: simply add them to the end of string_buf_ .

3 error handling

I set all possible regular match rules before the error action. If it can't match all the rules and have to match the error rules, I can make sure that something be wrong.I will call the err::ErrorMsg::Error() to record the error information.

4 end-of-file handling

It can be sovled by the ScannerBase class which was created by flexc++: when the text stream is end, the ActionType__ will be RETURN and the popStream() will return false. So the lex() will return 0. It can make the program exit the while loop in main(), and finish the program immediately.

5 other interesting features of my lexer

None.