Compilers

CMPT 432

Project One - 100 points

Project

Begin your compiler by writing a lexer that validates the input source code against the grammar found on our class web site at https://www.labouseur.com/courses/compilers/grammar.pdf. The source code will be read from a file for command-line compilers or an HTML text area element for browser-based compilers. [100 points]

Notes and Requirements

- Your lexer must lex multiple programs in sequence. Each program must be separated by the $ [EOP] marker.
- Do not attempt any parsing or semantic analysis for this project. No type checking, no scope checking, no symbol table, no CST… save it for future projects. Just get lex perfect. (Is that too much to ask?)
- The lexer is not as simple as our examples in class, so be careful. Be really careful.
- Provide both errors and warnings. Warnings are non-fatal mistakes or omissions that your compiler might correct. Forgetting to end the final program with $ is one example. Detecting unterminated comment blocks is another.
- When you detect an error, report it in helpful and excruciating detail including where it was found, what exactly went wrong, and how the programmer might fix it. I consider confusing, incomplete, or inaccurate error messages a serious (and intolerable) bug.
- Include verbose output functionality that traces the stages of the lexer. (An option for *GLaDOS mode* is good, but better *Yoda mode* is.)
- See the examples on the next page for details and ideas.

Other Requirements

You have to write this yourself. You may not use JavaCC, ANTLR, or any compiler compiler.

Create a plethora of test programs that cause as many different kinds of errors as you can in order to thoroughly test your code. (Think about code coverage and edge cases). Include several test cases that show it working as well. Write up your testing results (informally) in a document in your Git repository.

Your code must separate structure from presentation, be professionally formatted yet uniquely yours (show some personality), use and demonstrate best practices, and make me proud to be your teacher. [−∞ if not]

Hints

Remember the utility of comments and how much their presence and quality affect your professionalism and my opinion of your work.

Labs

Labs 1 and 2 focus on the components of this project. Completing those labs will help you with the project and also contribute to your preparation for the mid-term exam.

Submitting Your Work

Make **many** commits to GitHub. I do not want to see one massive "everything" commit when I review your code. (It's −∞ if you do that.) Commit early and often. 40 to 60 commits is a good goal for this project. Make sure your commit messages are descriptive, informative, and — if possible — entertaining.

E-mail me the URL to your **private** GitHub repository. (It must be a private repository. I will not accept anything else.) Remember to add me (Labouseur) as a collaborator. Please send this to me before the due date (see our syllabus).

Compilers

CMPT 432

Input:

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

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

Output to screen:

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

INFO  Lexer - Lexing program 2...
DEBUG Lexer - OPEN_BLOCK [ { ] found at (2:1)
DEBUG Lexer - OPEN_BLOCK [ { ] found at (2:2)
DEBUG Lexer - OPEN_BLOCK [ { ] found at (2:3)
DEBUG Lexer - OPEN_BLOCK [ { ] found at (2:4)
DEBUG Lexer - OPEN_BLOCK [ { ] found at (2:5)
DEBUG Lexer - OPEN_BLOCK [ { ] found at (2:6)
DEBUG Lexer - CLOSE_BLOCK [ } ] found at (2:7)
DEBUG Lexer - CLOSE_BLOCK [ } ] found at (2:8)
DEBUG Lexer - CLOSE_BLOCK [ } ] found at (2:9)
DEBUG Lexer - CLOSE_BLOCK [ } ] found at (2:10)
DEBUG Lexer - CLOSE_BLOCK [ } ] found at (2:11)
DEBUG Lexer - CLOSE_BLOCK [ } ] found at (2:12)
DEBUG Lexer - EOP [ $ ] found at (2:13)
INFO  Lexer - Lex completed with 0 errors

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

INFO  Lexer - Lexing program 4...
DEBUG Lexer - OPEN_BLOCK [ { ] found at (4:1)
DEBUG Lexer - I_TYPE [ int ] found at (4:36)
ERROR Lexer - Error:4:40 Unrecognized Token: @
DEBUG Lexer - CLOSE_BLOCK [ } ] found at (4:41)
DEBUG Lexer - EOP [ $ ] found at (4:42)
ERROR Lexer - Lex failed with 1 error(s)
```

```
INFO  Lexer - Lexing program 5…
DEBUG Lexer - OPEN_BLOCK [ { ] found at (5:1)
DEBUG Lexer - I_TYPE [ int ] found at (6:3)
DEBUG Lexer - ID [ a ] found at (6:7)
DEBUG Lexer - ID [ a ] found at (7:3)
DEBUG Lexer - ASSIGN_OP [ = ] found at (7:5)
DEBUG Lexer - ID [ a ] found at (7:7)
DEBUG Lexer - I_TYPE [ string ] found at (8:3)
DEBUG Lexer - ID [ b ] found at (8:10)
DEBUG Lexer - ID [ a ] found at (9:3)
DEBUG Lexer - ASSIGN_OP [ = ] found at (9:5)
DEBUG Lexer - ID [ b ] found at (9:7)
DEBUG Lexer - CLOSE_BLOCK [ } ] found at (10:1)
DEBUG Lexer - EOP [ $ ] found at (10:2)
INFO  Lexer - Lex completed with 0 errors
```