Lexical Analysis & Syntax Analysis

11712702 DengDun

Experimental procedure

1. define a struct node and its functions
2. create leaf node when doing lexical analysis
3. do bottom-up parsing in syntax analysis
4. error recovery

Problem & Solution

1. In the 'test_1_r10.spl', it need to report 'Missing Specifier' because it doesn't follow the rule `LC DecList StmtList RC`, but we need a error recover rule `LC DecList StmtList error` so it will report error at the end of `Stmt` because `a = 10;` can be recognized as `Dec` rather than `Def` without `Specifier`.

   It is not solved.

2. When meeting a lexical error, I didn't create a leaf node. So the syntax error will always happen after a lexical error.

   create a token `ERROR` and create corresponding rules to use it to replace some other token like `ID`

3. The definition of `CHAR` in `lex.l` cannot be matched correctly

   Some symbol need to add `\` before it (like `'`)

4. Cannot match `int` , `float`, `char` as `TYPE`

   Need to use `"int"`, `"float"`, `"char"`

5. Sometimes it will match wrong rule in syntax analysis

   Add `%right` and `%left` to define priority

6. Cannot return `node*` when match rule

   Define the type of `yylval`

7. Cannot match `TAB` and `EOL`

   Define `	` for `TAB` and `\n` for `EOL`

8. The biggest problem is that after I use the `stl` of `<list>` , the compiler tell me that it cannot find the `<list>` . Then I realize that it may not support `C++`.

   Write the part of `node` again without `stl`

Conclusion

From this project, I learn how to use flex and bison to do the easiest parsing, but it seems like that the given Syntax rule has some place different from the C language we use (Such as C can `define` `stmt` `define` but this language will report error). Some there are still a lot to design.