

*Lexical Analyzer*

Write a lexical analyzer for the CL language whose grammar you have previously received. Each call on "lex()" should return one token in the global structure "tok". tok.type should be set to an integer, indicating the type of the token (use a #define to give names to these types), and tok.id should be set to identify the specific instance of this token type. For <name>, tok.id should be the index of a location in the name table unique to that particular name. For <number>, tok.id should be the value of the number. For keywords and operators, the combination of tok.type and tok.id must be distinct for each, and can be encoded to suit your own style. Keywords should be entered into the symbol table initially, rather than processed one letter at a time by the FSM. For keywords, tok.type should identify the keyword uniquely, differentiating it from other values of tok.type, and tok.id should be 0.

Your program should run quickly, and avoid doing unneeded work such as storage allocation, and string copying, during processing of each token. lex() should be written so that, on each call, it assumes that one character beyond the end of the last token has already been read, and that character can be found in a global variable "Ch". lex() should take care to restore this condition on exit. lex() should also skip white-space and C style comments, and should keep track of the line number, for use in reporting errors.

To test your program, write a main program which calls lex() repeatedly, until EOF is detected. The main program should print the first 100 tokens read, one per line. Each line should contain 4 fields, separated by tabs: The fields (in order) are: (1) tok.type value; (2) tok.type mnemonic; (3) tok.id value; (4) tok.id string (for names only). At EOF, the main program should print a count of the number of ID occurrences, and the total number of tokens, including the EOF.

I suggest using the technique of allocating separate arrays to hold the various sorts of items your program will need. Each array's size should be a #defined name, to allow these sizes to be changed later. For guidance, I think that 1000 distinct names, and about 10000 total characters for all names and reserved words can be allocated. Possibly some 10000 distinct declared names will be needed.