

Principles of Compilers

Exercises: Lex

March 9, 2006

1. Describe the following languages that have been described as Lex regular expressions:
 - (a) $[+-][0-9]^+$
 - (b) $[A-Z][A-Z]^0,6$
 - (c) $a\(?x^*\?)$
 - (d) $0^*1^*(0|1)^*1^*0^*$
2. Write Lex regular expressions to capture:
 - (a) String literals in C (simplified). These are delimited by double quotes, and may not contain newline characters. They may contain a double quote if it is preceded by a single backslash `"\"`.
 - (b) Generalize the above definition of a string to include embedded newline sequences, i.e. a backslash followed by a newline character, which is interpreted that the string continues on the following line. Note there must be no characters between the backslash and the newline.
 - (c) Line comments in Ada. These start by `"--"`, and end with a newline character.
 - (d) Comments in Pascal. These are delimited by `"(*"` and `"*)"`. Note that Pascal comments do not nest (i.e. comments cannot include other comments), so we don't have to handle hierarchically embedded comments.
3. Construct using Lex a program for:
 - (a) translating all letter appearances in a text file into uppercase.
 - (b) replacing all integer numbers in a text file by its hexadecimal notation.
 - (c) eliminating all C-like comments from a text file.
 - (d) translating all Pascal-like keywords from a text file into German/Italian.
 - (e) reversing each line in a text file.

- (f) printing all HTTP tags defined in a text file, in alphabetical order.
 - (g) replacing in a text file all occurrences of "\$USERNAME" with the user's login name, of "\$TODAY" with the current date, and of "\$HOME" with the current home directory.
 - (h) recognising files (from the output of a ls command) according to their endings, and printing the name of the file and the corresponding type. Possible types are pictures files (".jpeg", ".tiff", ".png"), archive files (".rar", ".zip", ".gz", ".tar", ".tgz"), text files (".txt").
4. (due 23/3/2006) Write a scanner for your programming language, following the specifications you defined in the grammar. The scanner should recognise every keyword in the language (being case insensitive), initialise the symbol table for those nonterminals in the grammar that need attributes (for example identifiers), return the value of those nonterminals whose value will be needed by the compiler (for example number), and ignore comments. At this point, it is enough if you use a simple hash table for the symbol table.