CSCI 447 – Fall 2000
Scanner Generation with Lex

Professor: Muhammed Mudawwar
Due Date: Wednesday, October 18, 2000

Problem:

Write a Lex specification and use the Lex generator to construct a lexical analyzer for a language with the following lexical properties.

Lexical Conventions:

1. Single line comments begin with **--** (minus-minus) and end with the end of line. Multiple-line comments are surrounded by **(*** and ***)**. Comments cannot be nested and are ignored.
2. Blanks, tabs, and newlines are white space and should be ignored.
3. The following keywords are reserved: **and**, **array**, **boolean**, **char**, **do**, **else**, **end**, **false**, **function**, **if**, **in**, **integer**, **mod**, **not**, **of**, **or**, **out**, **program**, **real**, **return**, **then**, **true**, **while**. Keywords are not case sensitive.
4. Identifiers should begin with a letter, followed by zero or more letters, digits, or underscores '_'. Identifiers can be of any length and are not case sensitive.
5. The token **intconst** corresponds to an unsigned integer constant, which is a sequence of one or more digits. The token **realconst** corresponds to an unsigned real constant with at least one digit before and one digit after the decimal point, and with an optional exponent. The token **charconst** corresponds to a single character constant enclosed by single quotes. The following C escape sequences are permissible. **'\n'**, **'\t'**, **'\\'**, **'\0'**, **'\''**, and **'\"'** are the newline, tab, backslash, single quote, and double quote characters respectively. Other sequences should be rejected. The token **boolconst** corresponds to **true** and **false**.
6. The token **strconst** corresponds to a string literal. String literals are surrounded by double quotes. A string literal can be of any length, but should not span multiple lines. C escape sequences can be used within a string literal.
7. The token **relop** corresponds to the following operators: **= <> < <= > >=**
8. The token **addop** corresponds to the following addition operators: **+ -**
9. The token **mulop** corresponds to the following multiplication operators: *** / mod**
10. Other tokens that should be recognized are: **( ) [ ] , ; : :=**

Tables for Reserved Words, Identifiers, and String Literals

Use a table for the reserved words and lookup this table after matching an identifier. A hash function should be defined for the initialization and for the lookup of the table. The hash table may have more (unused) entries than the number of reserved words. The reserved words and their corresponding tokens should be entered into this table. Keywords should be converted to lowercase to simplify the lookup.

A second table (symbol table) should be used to enter identifiers. Only the first occurrence of an identifier should be entered. Look up the table first to decide whether an identifier should be entered. Hashing techniques should be used also for the fast search of this table. You should also maintain a string space with this table to store all identifiers. Allow the string space to grow dynamically. Identifiers should be converted to lower case before entering them to simplify the lookup.

A third table should be used to enter all string literals. This table is very similar to the identifier table. Only one instance of a string literal should be entered. Again, hashing

techniques should be used here for the fast lookup. A string space should also be used to store all the string literals. A string literal should be stripped from extra quotes and backslashes.

Error Handling

Write regular expressions to handle erroneous tokens. For instance, you will need regular expressions for non-terminated string literals ending with a newline character, for identifiers beginning with an underscore, and for real constants with no digits appearing before or after the decimal point. Print meaningful error messages for these lexical errors, but return correct tokens. Error messages should indicate the line number.

Token Attributes

You should keep track of the line number of each token as an attribute. For identifiers, a pointer to a symbol table entry is considered an attribute. For string literals, a pointer to a literal table entry is considered an attribute. For integer, real, char, and boolean literals, the matched string can be converted into an integer or floating-point number and becomes an attribute. For operators, a unique number identifying an operator is an attribute.

Output

The output is a list of tokens, lexemes, and corresponding line numbers. Error messages should appear in the proper place indicating the line at which the error has occurred. The contents of the symbol table and literal tables should be dumped at the end of the output file.

Main Function

The *main* function should accept the names of an input file and an output file specified on the command line. For example, if the executable file is called *scan* then one should type the following on a command line: **`scan infile outfile`**. The *main* function should call the *yylex* function repeatedly until the end-of-file token (token 0) is returned.

What you should submit

1. Put all your source files and executable file in one directory. Rename this directory as your *username*:

   `mv *your_assignment_directory username*`

   Create a tar file for your directory:

   `tar cvf *username*.tar *username*`

   Uuencode the tar file as follows (*username.tar* is written twice):

   `uuencode *username*.tar *username*.tar > *username*.uu`

   Mail the tar file to cs447:

   `mail –s "Lex assignment from *your full name*" cs447 < *username*.uu`

2. A report in which you discuss the regular expressions, their actions, the reserved word table, the symbol table, and the literal table, the hash function, and other details important to your solution. Test inputs, and outputs should also be appended to your report.

Grading

Your grade will be divided into the following components:

1. Correctness and output
2. Lex file, regular expressions, documentation of code.
3 Hashing tables and string spaces of reserved words, identifiers, and string literals
4 Error handling and reporting
5 Report