Gpp Interpreter Using Flex/Yacc

The “input.txt” is read line by line and firstly, evaluated by “gpp_lexer.l” to convert the elements in the lines into the keywords, operators, identifiers and values (only floating-values) in terms of lexical analysis. Then, using “gpp_interpreter.y”, the lines are evaluated by specified rules. Lastly, the analyzed line is printed to the “output.txt” with their calculated result.

TERMINAL SYMBOLS: Keywords, Operators and Identifiers as STRING and VALUEF as floating value.

NON-TERMINAL SYMBOLS
START -> INPUT | FDEF | EXIT
INPUT -> EXPI | EXPB | EXPLIST

EXPI
EXPI includes arithmetic operations (+, -, *, /), identifier, float values which is seperated by ‘f’ instead of ‘.’ (integer values are not evaluated because of the given CONCRETE_SYNTAX.pdf), variable definition with single float-values and multiple float-values as list, displaying/printing and if statements.

EXPB
EXPB includes logical operations (and, or, equal, less, not, true, false, nil) and its variations of floating-values and identifiers.

EXPLIST
EXPLIST includes list operations such as appending elements to a list, concatenating of two lists and definition of a list.

EXPLIST/VALUES
VALUES includes definition of list which is created by VALUEF.

FDEF
FDEF includes the definition of functions depending on their name as IDENTIFIER.

For definition of values, I am using an identifer table which is a struct array and struct contains the name and value/values of identifier. The identifiers that are used by various operations get or set the necessary information from/to that identifier table.

Gpp Interpreter Using LISP

The “input.txt” is read line by line and firstly, evaluated by “gpp_lexer.lisp” to convert the elements in the lines into the keywords, operators, identifiers and values (only floating-values) in terms of lexical analysis. Then, using “gpp_interpreter.lisp”, the lines are evaluated by specified rules. Lastly, the analyzed line is printed to the “output.txt” with their calculated result.

Function Explanations

gpp_interpreter: gpp_interpreter function reads the “input.txt” line by line and sends the line to the dfa function which is defined in “gpp_lexer.lisp” to convert the line into the terminal symbols. Then, process function is called to reduce the line and then, check-line function is called to printing the result depending on syntactic analysis.

process: process function calls reduce-tokens to reduce the terminal symbols into the nonterminal symbol such as expiarth and expb. Then, check-line is called to do syntactic analysis for making operation for determined type.

reduce-tokens: reduce-tokens function calls convert-expiarth and convert-expb funtions to reduce the specified terminal symbol sequences to expiarth and expb to make syntactic analysis easier and accurate.

convert-expiarth: convert-expiarth function reduces the arithmetic operations which are represented with multiple terminal symbols into single EXPIARTH nonterminal symbol depending on it satisfies the condition for an arithmetic operation.

calculate-expi: calculate-expi function makes the necessary operation to calculate the arithmetic operation by checking the operator in the selected subseqeunce.

convert-expb: convert-expiarth function reduces the logical operations which are represented with multiple terminal symbols into single EXPB nonterminal symbol depending on it satisfies the condition for an arithmetic operation.

calculate-expb: calculate-expi function makes the necessary operation to calculate the logical operation by checking the operator in the selected subseqeunce.

check-line: check-line function manages the syntactic analysis depending on the symbols in the selected lines. The arithmetic operations, logical operations, variable definitions, variable assignments, if statements are checked and the operation is done based on the satisfied condition for a line. Also, while loop and function definition are checked only for syntactically.

For definition of values, I am using an identifer table which is a map kind of a structue and structure contains the name and value/values of identifier. The identifiers that are used by various operations get or set the necessary information from/to that identifier table