Project

Lexer

I Build both the Lexical Analyzer (Lexer or Scanner) and Parser for the Pasc500 programming language. For some random input program (pasc500test1.p or pasc500test2.p). My Parser is called only once, to decide whether the program is syntactically correct, while Lexer is called repeatedly by Parser, each time the latter needs to advance to the next token of the program.

For the Lexer, I use the flex meta-tool where the tokens are encoded according to the Pasc500 programming language. Also, the information returned by Lexer (token code) will be returned via the yylval variable.

What you'll see:

1. Returns the token code it finds each time it's called.
2. Returns the names of the identifiers encountered.
3. Returns the values of the constants it encounters. For the non-numeric constants, it returns the string of the word it recognized.
4. Handle parsing errors by printing a message that includes the line number and the line of the input file where the error occurred. Try to continue by erasing the wrong character, and if this is not possible, then terminate it. (Max errors: 5)
5. For the purposes of this project, it should correctly print all the values it returns, in addition to any error messages.

Bison

For the Bison, I use the bison meta-tool which is easier to implement panic mode, a method of discovering the error and the parser discards input symbols one at a time.

What you'll see:

1. Recognize or reject the program based on the grammar of the language. One method is to use the precedence and associativity operators so that parsing leads to a unique syntax tree. (unambiguous grammar)
2. Handle parsing errors by printing messages that include the line number and the line of the input file where the error occurred. (Use the panic mode, max errors: 5)
3. Manage the scopes of the program according to the description of the language.
4. For the purposes of this project, it should correctly print all the values it returns at the end of each scope.