```
using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;
using System.ComponentModel.Design;
using System.IO;
using System.Reflection.Emit;

namespace KyleBushCompiler
{
    class Program
    {
        /*
         * CFG for Language Definition
         * <program> -> $UNIT <prog-identifier> $SEMICOLON <block> $PERIOD
         * <block> -> [<label-declaration>] {<variable-dec-sec>}* <block-body>
         * <block-body> -> $BEGIN <statement> {$SEMICOLON <statement>}* $END
         * <label-declaration> -> $LABEL <identifier> {$COMMA <identifier>}* $SEMICOLON
         * <variable-dec-sec> -> $VAR <variable-declaration>
         * <variable-declaration> -> {<identifier> {$COMMA <identifier>}* $COLON <type> $SEMICOLON}+
         * <statement>-> {<label> $COLON]}*
                        [
                            <variable> $ASSIGN (<simple expression> |
                            <string literal>) |
                            <block-body> |
                            $IF <relexpression> $THEN <statement> [$ELSE <statement>] |
                            $WHILE <relexpression> $DO <statement> |
                            $REPEAT <statement> $UNTIL <relexpression> |
                            $FOR <variable> $ASSIGN <simple expression> $TO <simple expression> $DO <statement> |
                            $GOTO <label> |
                            $WRITELN $LPAR (<simple expression> | <identifier> | <stringconst>) $RPAR
                        ]+
         * <prog-identifier> -> <identifier>
         * <statement> -> <variable> $ASSIGN <simple expression>
         * <variable> -> <identifier> [$LEFT_BRACKET <simple expression> $RIGHT_BRACKET]
         * <label> -> <identifier>
         * <relexpression> -> <simple expression> <relop> <simple expression>
         * <relop> -> $EQ | $LSS | $GTR | $NEQ | $LEQ | $GEQ
         * <simple expression> -> [<sign>] <term> {<addop> <term>}*
```

```
39          * <addop> -> $PLUS | $MINUS
40          * <sign> -> $PLUS | $MINUS
41          * <term> -> <factor> {<mulop> <factor> }*
42          * <mulop> -> $MULTIPLY | $DIVIDE
43          * <factor> -> <unsigned constant> | <variable> | $LPAR <simple expression> $RPAR
44          * <type> -> <simple type> | $ARRAY $LBRACK $INTTYPE $RBRACK $OF $INTEGER
45          * <simple type> -> $INTEGER | $FLOAT | $STRING
46          * <constant> -> [<sign>] <unsigned constant>
47          * <unsigned constant>-> <unsigned number>
48          * <unsigned number>-> $FLOAT | $INTTYPE
49          * <identifier> -> $IDENTIFIER
50          * <stringconst> -> $STRINGTYPE
51          */
52         static void Main(string[] args)
53         {
54             // Provided GOOD test file
55             //string inputFilePath = @"C:\projects\CS4100_Compiler_Design\TestInput\Part3BGOOD-1.txt";
56
57             // Provided BAD test file with syntax error
58             string inputFilePath = @"C:\projects\CS4100_Compiler_Design\TestInput\Part3B-BadTestfile1.txt";
59
60             // Provided BAD test file with lexical and syntax error
61             // string inputFilePath = @"C:\projects\CS4100_Compiler_Design\TestInput\BadProg2.txt";
62
63             // Initialize structures
64             ReserveTable reserveWords = InitializeReserveWordTable();
65             ReserveTable tokenCodes = InitializeTokenCodeTable();
66             SymbolTable symbolTable = new SymbolTable();
67
68             try
69             {
70                 // Initialize input file
71                 string[] fileText = InitializeInputFile(inputFilePath);
72
73                 // Initialize the Lexical Analyzer (Scanner)
74                 LexicalAnalyzer scanner = new LexicalAnalyzer();
75
76                 scanner.Initialize(fileText, symbolTable, reserveWords);
77                 bool echoOn = true;
```

```
 78
 79                  SyntaxAnalyzer parser = new SyntaxAnalyzer(scanner, tokenCodes, echoOn);
 80
 81                  scanner.GetNextToken(echoOn);
 82                  parser.TraceOn = false;
 83                  int val = parser.Program();
 84
 85                  symbolTable.PrintSymbolTable();
 86              }
 87              catch (Exception e)
 88              {
 89                  Console.WriteLine(e.Message);
 90              }
 91          }
 92
 93          /// <summary>
 94          /// Initializes the reserve table containing the token codes and mnemonics
 95          /// </summary>
 96          /// <returns>Reserve table containing the token codes and mnemonics</returns>
 97          static ReserveTable InitializeTokenCodeTable()
 98          {
 99              ReserveTable tokenCodes = new ReserveTable();
100
101              // Reserve Words
102              tokenCodes.Add("GOTO", 0);
103              tokenCodes.Add("_INT", 1);
104              tokenCodes.Add("__TO", 2);
105              tokenCodes.Add("__DO", 3);
106              tokenCodes.Add("__IF", 4);
107              tokenCodes.Add("THEN", 5);
108              tokenCodes.Add("ELSE", 6);
109              tokenCodes.Add("_FOR", 7);
110              tokenCodes.Add("__OF", 8);
111              tokenCodes.Add("WTLN", 9);
112              tokenCodes.Add("RDLN", 10);
113              tokenCodes.Add("_BEG", 11);
114              tokenCodes.Add("_END", 12);
115              tokenCodes.Add("_VAR", 13);
116              tokenCodes.Add("WHIL", 14);
```

```
117             tokenCodes.Add("UNIT", 15);
118             tokenCodes.Add("LABL", 16);
119             tokenCodes.Add("REPT", 17);
120             tokenCodes.Add("UNTL", 18);
121             tokenCodes.Add("PROC", 19);
122             tokenCodes.Add("DOWN", 20);
123             tokenCodes.Add("FUNC", 21);
124             tokenCodes.Add("RTRN", 22);
125             tokenCodes.Add("REAL", 23);
126             tokenCodes.Add("_STR", 24);
127             tokenCodes.Add("ARRY", 25);
128
129             // Other Tokens
130             tokenCodes.Add("_DIV", 30);
131             tokenCodes.Add("_MUL", 31);
132             tokenCodes.Add("_ADD", 32);
133             tokenCodes.Add("_SUB", 33);
134             tokenCodes.Add("LPAR", 34);
135             tokenCodes.Add("RPAR", 35);
136             tokenCodes.Add("SEMI", 36);
137             tokenCodes.Add("ASGN", 37);
138             tokenCodes.Add("__GT", 38);
139             tokenCodes.Add("__LT", 39);
140             tokenCodes.Add("GTEQ", 40);
141             tokenCodes.Add("LTEQ", 41);
142             tokenCodes.Add("__EQ", 42);
143             tokenCodes.Add("NTEQ", 43);
144             tokenCodes.Add("COMM", 44);
145             tokenCodes.Add("LBRC", 45);
146             tokenCodes.Add("RBRC", 46);
147             tokenCodes.Add("COLN", 47);
148             tokenCodes.Add("_DOT", 48);
149
150             // Identifiers
151             tokenCodes.Add("IDNT", 50);
152
153             // Numeric Constants
```

```
154             tokenCodes.Add("INTC", 51);
155             tokenCodes.Add("FLTC", 52);
156 
157             // String
158             tokenCodes.Add("STRC", 53);
159 
160             // Used for any other input characters which are not defined.
161             tokenCodes.Add("UNDF", 99);
162 
163             return tokenCodes;
164         }
165 
166         /// <summary>
167         /// Initializes reserve table with reserve words and token codes
168         /// </summary>
169         /// <returns>Reserve table with reserve words and token codes</returns>
170         static ReserveTable InitializeReserveWordTable()
171         {
172             ReserveTable reserveWords = new ReserveTable();
173 
174             // Token Codes
175             reserveWords.Add("GOTO", 0);
176             reserveWords.Add("INTEGER", 1);
177             reserveWords.Add("TO", 2);
178             reserveWords.Add("DO", 3);
179             reserveWords.Add("IF", 4);
180             reserveWords.Add("THEN", 5);
181             reserveWords.Add("ELSE", 6);
182             reserveWords.Add("FOR", 7);
183             reserveWords.Add("OF", 8);
184             reserveWords.Add("WRITELN", 9);
185             reserveWords.Add("READLN", 10);
186             reserveWords.Add("BEGIN", 11);
187             reserveWords.Add("END", 12);
188             reserveWords.Add("VAR", 13);
189             reserveWords.Add("WHILE", 14);
190             reserveWords.Add("UNIT", 15);
191             reserveWords.Add("LABEL", 16);
```

```
192             reserveWords.Add("REPEAT", 17);
193             reserveWords.Add("UNTIL", 18);
194             reserveWords.Add("PROCEDURE", 19);
195             reserveWords.Add("DOWNTO", 20);
196             reserveWords.Add("FUNCTION", 21);
197             reserveWords.Add("RETURN", 22);
198             reserveWords.Add("REAL", 23);
199             reserveWords.Add("STRING", 24);
200             reserveWords.Add("ARRAY", 25);
201
202             // Other Tokens
203             reserveWords.Add("/", 30);
204             reserveWords.Add("*", 31);
205             reserveWords.Add("+", 32);
206             reserveWords.Add("-", 33);
207             reserveWords.Add("(", 34);
208             reserveWords.Add(")", 35);
209             reserveWords.Add(";", 36);
210             reserveWords.Add(":=", 37);
211             reserveWords.Add(">", 38);
212             reserveWords.Add("<", 39);
213             reserveWords.Add(">=", 40);
214             reserveWords.Add("<=", 41);
215             reserveWords.Add("=", 42);
216             reserveWords.Add("<>", 43);
217             reserveWords.Add(",", 44);
218             reserveWords.Add("[", 45);
219             reserveWords.Add("]", 46);
220             reserveWords.Add(":", 47);
221             reserveWords.Add(".", 48);
222
223             return reserveWords;
224         }
225
226         /// <summary>
227         /// Reads all the text from the source file and stores each line as a seperate element in a string array.
228         /// </summary>
229         /// <param name="filePath">Path to the file to be read into memory</param>
```

```
230         /// <returns>The source text as a string array</returns>
231         static string[] InitializeInputFile(string filePath)
232         {
233             return File.ReadAllLines(filePath);
234         }
235     }
236 }
```