```
 1: #include <iostream>
 2: #include <fstream>
 3: #include <iomanip>
 4: #include "SyntaxAnalyzer.h"
 5:
 6: int main()
 7: {
 8:         const int COL_SIZE = 20;
 9:
10:         std::ifstream fin;                                      // Input file stream
11:         std::string inFile;                                     // Input file name
12:         std::vector<Lexer::Token> lineTokens; // List of lineTokens
13:         std::vector<Lexer::Token> tokens;
14:         std::stringstream *buffer;
15:         std::string line;
16:
17:         std::vector<std::string> files = {"test1.txt", "test2.txt", "test3.txt", "test4.txt"};
18:         std::ofstream out;
19:         out.open("output.txt");
20:
21:         for (std::string file : files)
22:         {
23:
24:                 // Open the file
25:                 fin.open(file.c_str());
26:
27:                 if (!fin)
28:                 {
29:                         out << "file not found" << std::endl;
30:                         continue;
31:                 }
32:
33:                 out << std::endl
34:                         << "RUNNING TEST CASE FILE \"" << file << "\"" << std::endl
35:                         << std::endl;
36:
37:                 // File has opened, instantiate the lexer.
38:                 Lexer *lexer = new Lexer();
39:
40:                 int lineNumber = 1;
41:
42:                 while (getline(fin, line))
43:                 {
44:                         buffer = new std::stringstream(line);
45:                         lineTokens = lexer->lex(*buffer, lineNumber);
46:
47:                         tokens.insert(tokens.end(), lineTokens.begin(), lineTokens.end());
48:
49:                         lineNumber++;
50:                 }
51:                 fin.close();
52:
53:                 SyntaxAnalyzer *syntaxAnalyzer = new SyntaxAnalyzer(tokens, out, false);
54:
55:                 try
56:                 {
57:                         // Run syntactical analysis
58:                         syntaxAnalyzer->Analyze();
59:                 }
60:                 catch (const SyntaxError &e)
61:                 {
62:                         out << std::endl
63:                                 << "ERROR: " << e.getMessage();
64:                 }
65:
66:                 tokens.clear();
67:                 out << syntaxAnalyzer->PrintAll();
68:
69:                 delete syntaxAnalyzer;
70:         }
71:
72:         out.close();
73:         std::cout << std::endl
74:                           << "EXECUTION HAS COMPLETED." << std::endl;
75:         std::cout << "Press enter to continue. . ." << std::endl;
76:         std::cin.get();
77:
78:         return 0;
79: }
```