

```
1: #ifndef SYNTAXANALYZER_H
2: #define SYNTAXANALYZER_H
3:
4: #include <fstream>
5: #include "Lexer.h"
6: #include "SymbolTable.h"
7:
8: class SyntaxError
9: {
10: public:
11:     // Constructor
12:     SyntaxError(std::string message, int lineNumber);
13:
14:     ~SyntaxError();
15:
16:     std::string getMessage() const;
17:
18: private:
19:     std::string message;
20:     int lineNumber;
21: };
22:
23: class SyntaxAnalyzer
24: {
25: public:
26:     // Constructor
27:     SyntaxAnalyzer(const std::vector<Lexer::Token> &tokens, std::ofstream &output, bool print = false);
28:     ~SyntaxAnalyzer();
29:
30:     // Begins the analysis process with the given tokens
31:     void Analyze();
32:
33:     std::string PrintAll();
34:
35: private:
36:     enum ErrorType
37:     {
38:         TYPE_MISMATCH,
39:         DUPLICATE_SYMBOL,
40:         UNDECLARED_VARIABLE
41:     };
42:
43:     void Rat18F();
44:     void OptFunctionDefinitions();
45:     void FunctionDefinitions();
46:     void Function();
47:     void OptParameterList();
48:     void ParameterList();
49:     void Parameter();
50:     void Qualifier();
51:     void Body();
52:     void OptDeclarationList();
53:     void DeclarationList();
54:     void Declaration();
55:     void IDs();
56:     void StatementList();
57:     void Statement();
58:     void Compound();
59:     void Assign();
60:     void If();
61:     void Return();
62:     void Print();
63:     void Scan();
64:     void While();
65:     void Condition();
66:     void Relop();
67:     void Expression();
68:     void Term();
69:     void Factor();
70:     void Primary();
71:     void Empty();
72:     void ExpressionPrime();
73:     void TermPrime();
74:     void Identifier();
75:     void Integer();
76:     void Real();
77:
78:     void error(ErrorType errorType, int lineNumber, std::string expected = "");
79:
80:     void getNextToken();
81:     void printCurrentToken();
82:
83:     const std::vector<Lexer::Token> &tokens;
84:     std::vector<Lexer::Token>::const_iterator it;
85:     Lexer::Token currentToken;
86:     bool print;
87:     std::ofstream &output;
88:     SymbolTable symbolTable;
89:     std::string *savedOp;
90:     std::string *savedType;
91:     Lexer::Token *save;
92:     std::ostringstream err;
93:     int errCount;
94:     bool isDeclaration;
95:     bool assign;
96: };
```

```
97:
98: #endif // SYNTAXANALYZER_H
```