

```
1: #ifndef SYNTAXANALYZER_H
2: #define SYNTAXANALYZER_H
3:
4: #include <fstream>
5: #include "Lexer.h"
6:
7: class SyntaxError
8: {
9: public:
10:
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:
27:     // Constructor
28:     SyntaxAnalyzer(const std::vector<Lexer::Token> &tokens, std::ofstream &output, bool print = false);
29:     ~SyntaxAnalyzer();
30:
31:     // Begins the analysis process with the given tokens
32:     void Analyze();
33:
34: private:
35:     void Rat18F();
36:     void OptFunctionDefinitions();
37:     void FunctionDefinitions();
38:     void Function();
39:     void OptParameterList();
40:     void ParameterList();
41:     void Parameter();
42:     void Qualifier();
43:     void Body();
44:     void OptDeclarationList();
45:     void DeclarationList();
46:     void Declaration();
47:     void IDs();
48:     void StatementList();
49:     void Statement();
50:     void Compound();
51:     void Assign();
52:     void If();
53:     void Return();
54:     void Print();
55:     void Scan();
56:     void While();
57:     void Condition();
58:     void Relop();
59:     void Expression();
60:     void Term();
61:     void Factor();
62:     void Primary();
63:     void Empty();
64:     void ExpressionPrime();
65:     void TermPrime();
66:     void Identifier();
67:     void Integer();
68:     void Real();
69:
```

```
70:  void getNextToken();
71:  void printCurrentToken();
72:
73:  const std::vector<Lexer::Token> &tokens;
74:  std::vector<Lexer::Token>::const_iterator it;
75:  Lexer::Token currentToken;
76:  bool print;
77:  std::ofstream &output;
78: };
79:
80: #endif // SYNTAXANALYZER_H
```