Role of a Parse Table in Syntax Analysis

In syntax analysis, a parse table plays an important role in guiding the parsing process. It helps the parser determine which grammar production should be applied based on the current non-terminal and the next input symbol.

For predictive (LL) parsers, the parse table is constructed using the **FIRST** and **FOLLOW** sets of the grammar. During parsing, the table is consulted to:

- Apply the correct production rule
- Validate whether the input string follows the grammar
- Detect syntax errors when no valid rule exists

Overall, the parse table ensures that the parser processes the input systematically and verifies the syntactic correctness of the program.

C++ Program: Counting Operators in an Expression

```
#include <string>

using namespace std;

int countOperators(string expression) {
    int count = 0;
    for (char ch : expression) {
        if (ch == '+' || ch == '-' || ch == '*' || ch == '/') {
            count++;
        }
    }
    return count;
}
```

This function scans the expression character by character and checks for arithmetic operators (+, -, *, /). Each time an operator is found, the counter is increased. The final count represents the total number of operators in the expression.

problem Solving: Parse Tree Construction

Given Grammar: S → 0S1 | 01
Input String: 0011

Derivation Steps

S

→ 0S1

→ 0(01)1

→ 0011

Parse Tree

The string 0011 is derived by applying the production S → 0S1. The inner non-terminal S is replaced by 01, which results in the valid string 0011. The parse tree visually represents how the grammar generates the input string.

Conclusion

This assignment demonstrates an understanding of syntax analysis concepts, grammar-based parsing, and basic compiler-related programming in C++. The answers show how theoretical concepts such as parse tables and grammars are applied in practical problem-solving and implementation.