# Description

An LL(1) parser acts as a top-down parser capable of parsing strings from a language created by an LL(1) grammar. Given the current lookahead symbol, an LL(1) grammar is a context-free grammar in which each non-terminal symbol has at most one production rule that can be used to extend it.

The LL(1) parser in C works as follows:

1. It removes recursion using the provided production rules first.
2. The FIRST and FOLLOW sets are then computed for every non-terminal symbol.
3. It then builds a parsing table using the FIRST and FOLLOW sets.
4. Finally, it parses the supplied string using the parsing table.

All CFG parsers operate by traversing the input string's derivation tree. The LL(1) parser is a subset of the CFG parser that uses the parsing table to decide how to explore the derivation tree.

# Connection between the solution and the concept learned

Top-down parsing is a parsing approach that begins at the top of a parse tree and works its way down, incorporating non-terminal symbols to production rules. Top-down parsing algorithms are frequently recursive, which means they call themselves to parse various parts of the input string.

The parser begins with a prediction about the composition of the input string and then attempts to validate that prediction by transforming non-terminal symbols to production rules. If the parser fails to verify its prediction, it goes back and tries again.

The concept of top-down parsers is used to develop the LL1 Parser that is therefore capable of parsing strings from a language created by an LL(1) grammar.