**Peanut Compiler Syntax Analysis - CS323 Documentation**

1. **Problem Statement**

After lexical analysis, the next step of the compilation process is syntactical analysis. The syntax analyzer must read tokens and be able to tell if they are syntactically correct. The expected syntax for Rat23F source code is defined in productions, listed in the Rat23F documentation. Additionally, because of the limitations of the most easily implementable syntax parsing algorithm, Recursive Descent Parser (RDP), the Rat23F syntax rule set cannot be used in its original form since it contains left recursion and backtracking, which causes the parser to enter infinite loops.

1. **Usage**

***This is a command-line utility.***

1. Using the Command Prompt, navigate to the directory that *peanut-compiler.exe* is located in.

2. Run *peanut-compiler.exe* while passing the path to the Rat23F source file to process as the first argument.

3. See the output in the console to view the results.

A screenshot of a computer screen

Description automatically generated

* To view more information about the accepted arguments, run *peanut-compiler.exe* without any arguments.

A screenshot of a computer program

Description automatically generated

1. **Program design**

The source code can be found at <https://github.com/jiink/peanut-compiler/tree/main/src/>

This program’s syntax analyzer implements a Recursive Descent Parser that operates on a modified version of the Rat23F production. The productions are modelled as function calls; each nonterminal symbol has its own function, making for 41 function definitions. These modified productions are outlined in *rules.docx*.

1. **Limitations**

* Identifiers, reals, and integers are limited to 500 characters.
  + This limit is arbitrary and is placed simply to prevent any possibility of infinite loops in the finite state machines.
* Syntax analysis will halt upon finding the first syntax error.

1. **Shortcomings**

None known.