Complete Grammar of CBaby Language

Predefined Productions Used in Grammar

The following are the defined productions that are used in grammar for parser design. Note that these are already defined (in Lex implementation documentation), and are just listed down for readability:

letter -> a | b | c | … | z | A | B | C | … | Z

digit -> 0 | 1 | 2 | … | 9

spC -> ~ | ` | ! | @ | $ | % | & | * | (|) | { | } | [|] | + | = | _ | - | \ | / | < | > | . | , | ” | ’ | space | : | ;
| ? | | |

TD -> Integer | char

VDO -> :

ID -> letter (letter | digit)*

NC -> digit (digit)*

LC -> ‘letter’

STR -> “ (letter | digit | spC | ^)* “

ReOp -> (< | (^ | =)) | (> | (^ | =)) | (==) | (/=)

AO -> :=

IO -> >>

newLine -> \n

SLC -> /* (letter | digit | spC | ^)* */

MLC -> /* (letter | digit | spC | newLine | ^)* */

CFG for CBaby Language

Now, our task is to design a parser for the “CBaby” language. First, we need a complete Context Free Grammar (CFG) for its implementation. Following is the complete grammar of “CBaby” language and this will be used to implement its parser.

```
Start -> Function Start | forStatement Start | ^

Function -> func TD VDO ID ( forParam ) { forStatement }

forParam -> sendParam | ^

sendParam -> Param nextParam

nextParam -> , Param nextParam | ^

Param -> TD VDO ID

Variable -> TD VDO ID VariableDelimiter

VariableDelimiter -> ; | , nextVariable

nextVariable -> Variable | ID VariableDelimiter

forStatement -> Statement  forStatement | ^

Statement -> if Condition VDO { forStatement } ElifOrElse
             | while Condition VDO { forStatement }
             | print  ( OutputOptions ) ; | println  ( OutputOptions ) ; | In IO ID InputDelimiter
             | ret FCParam ; | ret ; | AssignmentStatement
             | Variable | FunctionCall ; | SLC | MLC

ElifOrElse -> elif Condition VDO { forStatement } ElifOrElse | goElse

goElse -> else { forStatement } | ^

forNewLine -> ln | ^

OutputOptions -> ID moreOutput  | LC moreOutput | NC moreOutput |
                 STR moreSTROutput | Expression moreOutput

moreOutput -> , OutputOptions | ^
moreSTROutput -> + OutputOptions | ^
```

InputDelimiter -> ; | , nextInput

nextInput -> ID InputDelimiter

AssignmentStatement -> ID AO SelectOption ; | Param AO SelectOption ;

SelectOption -> FunctionCall | FCParam

FunctionCall -> ID (forFCParam)

forFCParam -> sendFCParam | ^

sendFCParam -> FCParam nextFCParam

nextFCParam -> , FCParam nextFCParam | ^

FCParam -> ID | NC | LC | Expression

Condition -> Expression ReOp Expression | Expression == Boolean | Expression /= Boolean
| Boolean

Boolean -> true | false | 0 | 1

Expression -> addOperand addOperatorPart1

addOperatorPart1 -> + addOperand addOperatorPart1 | - addOperand addOperatorPart1 | ^

addOperand -> Operand addOperatorPart2

addOperatorPart2 -> * Operand addOperatorPart2 | / Operand addOperatorPart2 | ^

Operand -> ID | NC | LC