TOC Assignment 2

Akshat Oke 2020A7PS0284H
Aaditya Rathi 2020A7PS2191H
Sankalp Kulkarni 2020A7PS1097H
Shreyas Dixit 2020A7PS2079H
Chirag Gadia 2020A7PS1721H

December 2022

Grammar for Simple C

Start symbol is ‘MainProgram‘

⟨*MainProgram*⟩ ::= ⟨*Declarations*⟩ ⟨*Program*⟩

⟨*Declarations*⟩ ::= ‘`int`’ ⟨*identifier*⟩ [‘`,`’ ⟨*identifier*⟩]* ‘`;`’

⟨*Program*⟩ ::= ⟨*statement*⟩*

⟨*statement*⟩ ::= (⟨*assignment*⟩
| ⟨*read-statement*⟩
| ⟨*write-statement*⟩
| ⟨*for-statement*⟩
) ‘`;`’

⟨*assignment*⟩ ::= ⟨*identifier*⟩ ‘`=`’ ⟨*expression*⟩

⟨*expression*⟩ ::= ⟨*equality*⟩

⟨*equality*⟩ ::= ⟨*comparison*⟩ [‘`==`’ ⟨*comparison*⟩]?

⟨*comparison*⟩ ::= ⟨*term*⟩ [‘`>`’ ⟨*term*⟩]?

⟨*term*⟩ ::= ⟨*factor*⟩ [(‘`+`’ | ‘`-`’) ⟨*factor*⟩]*

⟨*factor*⟩ ::= ⟨*unary*⟩ [(‘`*`’ | ‘`/`’) ⟨*unary*⟩]*

⟨*unary*⟩ ::= (‘`-`’ ⟨*unary*⟩) | ⟨*primary*⟩

⟨*primary*⟩ ::= ⟨*integer*⟩ | ⟨*identifer*⟩ | ‘`(`’ ⟨*expression*⟩ ‘`)`’

⟨*read-statement*⟩ ::= ‘`read`’ ⟨*identifier*⟩

⟨*write-statement*⟩ ::= ‘`write`’ (⟨*integer*⟩ | ⟨*identifer*⟩)

⟨*for-statement*⟩ ::= ‘`for`’ ‘`(`’ ⟨*expression*⟩ ‘`;`’ ⟨*expression*⟩ ‘`;`’ ⟨*expression ‘)’ ¡block*⟩

⟨*block*⟩ ::= ‘{’ ⟨*statement*⟩* ‘}’

⟨*identifier*⟩ ::= -⟨*keyword*⟩ ⟨*alpha*⟩+

⟨*integer*⟩ ::= ⟨*digit*⟩+

About the implementation

Types

Since the document mentions nothing about types, this implementation assumes 'Simple C' to be strongly typed, with two types.

There are two types of expressions: number and boolean. Only a boolean expression can be used in the condition clause of the for statement.
Printing a boolean variable will print 'true' or 'false' correspondingly. Booleans cannot be operated upon by arithmetic operations.

Uninitialized variables

The ¡Declarations¿ rule declares variables that can be used in the program. Variables have been implemented with a hashmap. Referencing a variable before assigning a value to it causes an 'Undefined variable' runtime error.

It was possible to implicity give variables an initial value of 0, but this may lead to unreliable code.

Remarks

All three clauses in the for statement have no side effects. This means they can be executed any number of times and in any order during the loop execution. Although it seemed that I could get away easily by not using the AST for interpreting, the additional "rewing" functionality required in the scanner was better left alone.