SER 502 Project Milestone 1

Team 2:
Jacob Jose
Michael Kangas
Boan Li
Tahir Pervez
James Thayer

Language Name
Brewless

Grammar
Program -> Block | Command
Block -> Command; Block | Command

Command -> Declaration | Assignment | Conditional | For_loop | While_loop | Ternary | Print

Declaration -> Type Identifier
Type -> int | String | boolean
Assignment -> Identifier = Expression

Expression -> Addition | Subtraction | Multiplication | Division | (Expression) | Identifier | Integer | String | Boolean

Integer -> Digit Integer | Digit
Identifier -> Letter Identifier | Letter
Digit -> 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
Letter -> a | b | c | … | z | A | B | C | … | Z

Boolean -> true | false | Expression == Expression | not (Boolean) | Boolean and Boolean | Boolean or Boolean
String -> string Identifier = “Identifier”

Multiplication -> Expression * Expression
Division -> Expression / Expression
Addition -> Expression + Expression
Subtraction -> Expression - Expression

Conditional -> if (Boolean) { Block } | if (Boolean) { Block } else { Block }
Ternary -> Boolean ? Expression : Expression

For_loop -> for (Assignment; Boolean; Loop_update) { Block } | for (Identifier in Range (Integer,Integer)) { Block }

While_loop -> while (Boolean) { Block } | do { Block } while (Boolean)

Loop_update -> Assignment | Increment | Decrement
Increment -> Identifier++ | ++Identifier
Decrement -> Identifier-- | --Identifier

Print -> print (Identifier)

Design

The design of the programming language will be kept similar to Java/C and is supposed to be an imperative language. We will be using lex to tokenize the program. We will be producing our lexical analyzer and parse tree using DCG, and creating our semantic analyzer and runtime environment with Python. Below are explanations for the different grammar rules.

Program: Baseline starting point of a program, can contain a block of code or a single command.

Block: Section of code within the program, contains at least one command and potentially more commands and blocks.

Command: Basic operation of the language, are declarations, assignments, conditionals, loops, a ternary operation, or a print statement.

Declaration: A type of command, is a type and an identifier used to declare a variable.

Type: A generic term that refers to "int", "String", or "bool" variables.

Assignment: A command to assign an identifier to some expression.

Expression: A generic term for program statements. Contains arithmetic operations as well as identifiers and numbers.

Multiplication: The multiplication operation of two expressions

Division - Mathematical Division.

Addition: Addition operation for two expressions.

Subtraction: Subtraction operation for two expressions.

Identifier: Is a variable with a value that may be changed. These contain either a single character or "letter", or potentially multiple letters. Stores data.

Digit: Actual digits from 0 to 9. Used to make up integer values.

Integer: Any whole number.

Letter: Equivalent to characters in other languages. All capital and lowercase letters.

Boolean: True or false logic, used in conditional statements.

String: Specific identifier not used specifically for variable name but stored as data.

Conditional: The actual if , else statements within the language, does not include ternary expressions. Works with Booleans and can contain new blocks of code

Ternary: Syntactic sugar for a conditional with a return, where left is true and right if false.

For_loop: Two different for loop types including traditional for loop and an in range for loop. Traditional for loop uses a loop_update value.

While_loop: Two forms of while loops, a traditional while loop and a do while loop. Both operate on booleans and can contain blocks of code.

Loop_update: incrementer for a for loop, can contain an increase or decrease increment

Increment: The increase increment increases a value by 1 and returns it.

Decrement: The decrease increment decreases a value by 1 and returns it.

Print: A classic print statement, prints an identifier.

Control flow syntax:

```
if (Boolean) {
  Block
}

if (Boolean) {
  Block
} else {
  Block
}
```

Ternary:

(Boolean ? Expression (block ran if true) : Expression (block ran if false))

For Loops:

```
for (Assignment; Boolean; Loop_update) {
  Block
}

for (Identifier in Range (Integer,Integer) {
  Block
}
```

While Loops:

```
while (Boolean) {
  Block
}
do {
  Block
} while (Boolean)
```

Print:

print(“example”)

Comment syntax: The symbol ~ will be used for comments

Links

GitHub Repo