Zoo Programming Language

Language Specification:

1. Language Definition
    a. Alphabet:
        i. Upper (A-Z) and lower case letters (a-z) of the English alphabet
        ii. Underline character '_'
        iii. Decimal digits (0-9)
    b. Lexic:
        i. Special symbols, representing:
            1. operators: `+` `-` `*` `/` `%` `<>` `<<` `>>` `!=` `<-` `<=` `>=`
            2. Separators: `( )` `{ }` `[ ]` `;` `(space)`
            3. Reserved words:

                zoo cat cow dog wolf owl penguin cheetah bee wasp

        ii. Identifiers
            1. A sequence of letters and digits and underline character, such that the first character is a letter; the rule is:

```
identifier = letter | letter{letter | digit | underline}
letter = "A" | "B" | ... | "Z" | "a" | "b" | ... | "z"
digit = "0" | "1" | ... | "9"
underline = "_"
```

        iii. Constants
            1. Integer - rule:

```
number_constant = [(+ | -)] non_zero_number | "0"
non_zero_number = non_zero_digit{digit}
non_zero_digit = "1" | "2" | "3" | ... | "9"
```

            2. Character - rule:

```
character = 'letter' | 'digit' | 'symbol'
```

```
symbol = "_" | "-" | " "
```

3. String - rule:

```
const_string = "string"
```

```
string = char{string}
```

```
char = letter | digit | symbol
```

c. Syntax:

The words - predefined tokens are specified between " and "

i. Syntactical rules:

```
program = "zoo" compound_statement "zoo"
```

```
compound_statement = (declaration | statement) ";" [compound_statement]
```

```
declaration = primitive_type identifier ["<-" expression]
```

```
constant = number_constant | character | const_string
```

```
primitive_type = "cat" | "bee" | "wasp"
```

```
array_type = "cow" "<" primitive_type ">" "[" number_constant "]" identifier
```

```
statement = simple_statement | structured_statement
```

```
simple_statement = assign_statement | io_statement
```

```
io_statement = read_statement | write_statement
```

```
read_statement = "owl" "(" identifier ")"
```

```
write_statement = "penguin" "(" identifier ")"
```

```
assign_statement = id "<-" expression
```

```
expression = term [("+" | "-")expression]
```

```
term = factor [("*" | "/" | "%") term]
```

```
factor = identifier | number_constant | "(" expression ")"
```

```
structured_statement = if_statement | while_statement
```

```
if_statement = "dog" "(" relational_expression ")" "{" compound_statement "}" | "dog" "{" compound_statement "}" "wolf" "{" compound_statement "}"
```

```
while_statement = "cheetah" "(" relational_expression ")" "{" compound_statement "}"
```

```
relational_expression = expression relational_operator expression
```

```
relational_operator = << | >> | <> | != | <= | >=
```