Bagamaspad, Anne Marielle CMPILER Enomoto, Yuki

Choa, Shaila Mae Tan, Jaylica Anne

1. **Lexical Specifications**

|  |  |  |  |
| --- | --- | --- | --- |
| **Token Type** | **Token Type Description** | **Regular Expression** | **Regular Expression Description** |
| Main Keyword |  | zoo |  |
| Integer Keyword |  | sheep |  |
| Float Keyword |  | otter |  |
| String Keyword |  | snake |  |
| Character Keyword |  | worm |  |
| Boolean Keyword |  | boo |  |
| Array Keyword |  | spider |  |
| Integer Literal |  | [-+]? [0-9]+ |  |
| Float Literal |  | [-+]?[0-9]+.[0-9]+ |  |
| String Literal |  | ~[0-9A-Za-z\s]\*~ |  |
| Character Literal |  | `[0-9A-Za-z]` |  |
| Boolean Literal |  | diurnal/ nocturnal |  |
| Assignment Operator |  | = |  |
| Terminator |  | <3 |  |
| Printf Keyword |  | purr |  |
| Scanf Keyword |  | rawr |  |
| If Keyword |  | dog |  |
| Else Keyword |  | cat |  |
| Else If Keyword |  | catdog |  |
| For Loop Keyword |  | foodchain |  |
| While Loop Keyword |  | run |  |
| Do While Loop Keyword |  | move |  |
| Open Parenthesis |  | ( |  |
| Close Parenthesis |  | ) |  |
| Open Bracket |  | [ |  |
| Close Bracket |  | ] |  |
| Opening Braces |  | e-worm | Also used as opening code block |
| Closing Braces |  | f-worm | Also used as closing code block |
| += operator |  | arfthis |  |
| -= operator |  | sssthis |  |
| * operator |  | arf |  |
| * operator |  | sss |  |
| x operator |  | meow |  |
| / operator |  | dook |  |
| % operator |  | mooodulo |  |
| ^ operator |  | powpow |  |
| Increment |  | mate |  |
| Decrement |  | prey |  |
| Function Identifier |  | func[A-Za-z]\* |  |
| Next Line Keyword |  | \n |  |
| Null |  | extinct |  |
| Return Keyword |  | back |  |
| Less Than operator |  | < |  |
| Less Than Equal To operator |  | <= |  |
| Greater Than operator |  | > |  |
| Greater Than Equal To operator |  | >= |  |
| Equal To operator |  | == |  |
| Not Equal To operator |  | != |  |
| And |  | && |  |
| Or |  | || |  |
| Not |  | ! |  |
| Separator |  | , |  |
| Comment Block |  | #[0-9A-Za-z]+# |  |
| Variable Identifier |  | [A-Za-z]+ | Variable Identifier |

**B. Context Free Grammar**

|  |  |
| --- | --- |
| **Production Rule** | **Description of Non-Terminals and Production Rule** |
| ***Variable Declaration*** | |
| var\_dec -> datatype var identifier **|** datatype var identifier multivar\_dec | statement used for declaring a variable of a particular datatype |
| var\_init -> datatype var identifierassign operatorto\_assign terminator | declare a variable with an initial value |
| multivar\_dec -> next\_var terminator | declare at least one variable of a datatype |
| next\_var -> separator var identifier next\_var | E | used for declaring the nth variable in a multi-variable declaration |
| datatype -> int\_keyword|float\_keyword|string\_keyword | char\_keyword | the different datatypes |
| arr\_dec -> datatype array\_keyword var\_identifier open bracket literal close bracket terminator | declare an array |
| lit\_var -> literal | var identifier |  |
| ***Assignment Statements*** | |
| var\_assign -> var identifierassign operatorlit\_var terminator | assign a value to a declared variable |
| add\_sub -> += operator | -= operator | add or subtract operators to the first variable |
| inc\_dec -> increment| decrement |  |
| incdec\_assign -> var identifierinc\_decterminator | Increment and decrement the value of a variable |
| addsub\_assign -> var identifieradd\_sublit\_var terminator | add/sub and assign to declared variable |
| literal -> int literal|float literal|string literal| char literal|expr | the different types of literals (may be grouped by parentheses) |
| var\_assign2 -> var identifierassign operatorexpr terminator | assign a value of a given expression to the variable |
| ***Arithmetic Expressions*** | |
| expr -> expr op expr | Open Parenthesis expr Close Parenthesis | literal | var identifier | E | mathematical expressions |
| op -> + operator | - operator| x operator |/ operator| % operator | ^ operator | and | or | +, -, \*, /, %, ^, &&, || operators |
| ***Conditional Statements*** | |
| cond\_op -> Less Than operator | Less Than Equal To operator | Greater Than operator | Greater Than Equal To operator | Equal To Operator | Not Equal To operator | <, <=, >, >=, ==, != operations |
| cond\_expr -> lit\_var cond\_op lit\_var | expression to be evaluated by if-else |
| conditional -> if\_keywordOpen Parenthesis cond\_expr expr Close Parenthesis code\_block next\_conditional | first condition |
| next\_conditional ->elseif\_keyword Open Parenthesis cond\_expr Close Parenthesis code\_block | next\_conditonal2 | E | succeeding conditions |
| next\_conditional2 -> else\_keywordcode\_block | E | last case without condition |
| ***Loop Statements*** | |
| for\_loop -> For Loop KeywordOpen Parenthesis cond\_expr Close Parenthesis Opening Bracescode\_blockClosing Braces | for loop |
| while\_loop -> While Loop KeywordOpen Parenthesis cond\_expr Close Parenthesis Opening Bracescode\_blockClosing Braces | while loop |
| dowhile\_loop -> Do While Loop KeywordOpening Bracescode\_blockClosing Braceswhile loop keyword Open Parenthesis cond\_expr Close Parenthesis terminator | do while loop |
| ***Function Declaration & Call*** | |
| param -> datatype var identifier | var identifier | param seperator param | expr | E | parameter |
| func\_def -> datatype func\_name Open Parenthesis param Close Parenthesis code\_block | definition of a function |
| func\_call -> func\_name Open Parenthesis param Close Parenthesis terminator | function call |
| ***Code Block*** | |
| code\_block -> Opening Braces statement Closing Braces |  |
| statement -> var\_dec | var\_init | arr\_dec | var\_assign | incdec\_assign | addsub\_assign | var\_assign2 | conditional | loop | func\_def | func\_call | all possible statements in a code block |
| ***Main*** | |
| start -> datatype Main Keyword Open Parenthesis Close Parenthesis code block |  |

You are missing production rules that cover arrays – Sir Thomas

Addition:

A.

1. void – neuter

1. add printf, scanf