**Alphabet:**

Upper (A-Z) and lowercase letters (a-z) of the English alphabet

Decimal digits (0-9)

**Lexic:**

a. Special symbols, representing:

- operators

- arithmetic: + - \* / %

- relational: > >= < <= = !=

- logical: and or not

- assignment: is

- separators ( ) [ ] { } ; space ‘’

- reserved words: inty booly ify elseify elsy loopy to begin\_appy end\_appy

b. Identifiers - a sequence of letters and digits, such that the first character is a letter

identifier = letter{letter | digit}

letter = “A” | ‘B” | ... | “Z” | “a” | “b” | ... | “z”

digit = “0” | “1” | ... | “9”

nz\_digit = “1” | ... | “9”

c. Constants

- integer = “0” | [“ + “ | “ – “] nz\_digit { “0” | nz\_digit }

- bool = “true” | “false”

**Tokens:**

|  |  |
| --- | --- |
|  |  |
| identifier | 0 |
| constant | 1 |
| begin\_appy | 2 |
| end\_appy | 3 |
| inty | 4 |
| booly | 5 |
| ify | 6 |
| elsify | 7 |
| elsy | 8 |
| loopy | 9 |
| to | 10 |
| ( | 11 |
| ) | 12 |
| [ | 13 |
| ] | 14 |
| { | 15 |
| } | 16 |
| ; | 17 |
| , | 18 |
| is | 19 |
| > | 20 |
| >= | 21 |
| = | 22 |
| != | 23 |
| < | 24 |
| <= | 25 |
| + | 26 |
| - | 27 |
| \* | 28 |
| / | 29 |
| % | 30 |
| and | 31 |
| or | 32 |
| not | 33 |

**Syntax:**

program = “begin\_appy” { declaration } stmt\_list “end\_appy”

declaration = type identifier {“,” identifier}“;”

basetype = “inty” | “booly”

array\_declaration = arry[basetype] identifier“[” nz\_digit { “0” | nz\_digit } “]”“;”

type = basetype | array\_declaration

stmt = simple\_stmt | struct\_stmt

simple\_stmt = (assignment\_stmt | io\_stmt) ";"

struct\_stmt = compound\_stmt | if\_stmt | loop\_stmt

compound\_stmt = “{” stmt\_list “}”

stmt\_list = stmt { stmt }

expression = [“not”](term | expression operation expression)

operation = “+” | “-“ | “\*” | “/” | “%” | “and” | “or"

term = identifier | integer | bool | identifier"["identifier"]"

assignment\_stmt = identifier “is” expression

io\_stmt = (“pickup” | “sparkle”)(identifier)

if\_stmt = “ify” condition “{” stmt “}” { “elsify” condition “{” stmt “}” } [“elsy” “{” stmt “}”]

loop\_stmt = “loopy” loop “{” stmt “}”

condition = “(” expression relation expression “)”

loop = “(” inty identifier“;” expression “to” expression“;” integer “)”

relation = “<” | “<=” | “=” | “!=” | “>=” | “>” | “and” | “or”