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” 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”
```