# 语法

## 带行数版：

1. program: global\_area main\_func
2. global\_area: global\_area const\_declaration ';'
3. | global\_area var\_declaration ';'
4. | global\_area func\_declaration
5. | global\_area class\_declaration
6. |
7. const\_declaration: LET const\_list
8. const\_list: const\_list ',' const\_expr
9. | const\_expr
10. const\_expr: name '=' const\_value
11. const\_value
12. : INTEGER
13. | FLOAT
14. | DOUBLE
15. | CHAR
16. | BOOLEAN
17. | STRING
18. var\_declaration: VAR var\_list
19. para\_list: var\_list
20. var\_list: var\_list ',' var\_expr
21. | var\_expr
22. var\_expr: name '=' expression
23. | name '=' lambda\_expression
24. | name ':' type\_declaration
25. | name
26. type\_declaration: var\_type
27. | array\_type\_declaration
28. | func\_type
29. array\_type\_declaration: var\_type '[' INTEGER ']'
30. func\_type: FUNCTION
31. var\_type: TYPE\_INT
32. | TYPE\_INT\_POINTER
33. | TYPE\_INT\_64
34. | TYPE\_INT\_64\_POINTER
35. | TYPE\_CHAR
36. | TYPE\_CHAR\_POINTER
37. | TYPE\_FLOAT
38. | TYPE\_FLOAT\_POINTER
39. | TYPE\_DOUBLE
40. | TYPE\_DOUBLE\_POINTER
41. | TYPE\_BOOL
42. | TYPE\_BOOL\_POINTER
43. func\_declaration: FUNCTION name '(' para\_list ')' compound\_statement
44. | FUNCTION name '(' para\_list ')' OPER\_PTR var\_type compound\_statement
45. main\_func: FUNCTION MAIN '(' ')' OPER\_PTR TYPE\_INT compound\_statement
46. statement\_list: statement\_list statement
47. |
48. statement: compound\_statement
49. | branch\_statement
50. | for\_statement
51. | jump\_statement ';'
52. | assign\_statement ';'
53. | var\_declaration ';'
54. | const\_declaration ';'
55. | name '(' expression\_list ')' ';'
56. compound\_statement: '{' statement\_list '}'
57. branch\_statement: IF '(' expression ')' compound\_statement
58. | IF '(' expression ')' compound\_statement ELSE compound\_statement
59. for\_statement: WHILE '(' expression ')' compound\_statement
60. | FOR name IN '[' expression ',' expression ',' expression ']' compound\_statement
61. jump\_statement: JUMP\_BREAK
62. | JUMP\_CONTINUE
63. | JUMP\_RETURN
64. | JUMP\_RETURN expression
65. expression: expression OPER\_OR expression\_or
66. | expression\_or
67. expression\_or: expression\_or OPER\_AND expression\_and
68. | expression\_and
69. expression\_and: expression\_and OPER\_EQ expr
70. | expression\_and OPER\_NE expr
71. | expression\_and OPER\_GT expr
72. | expression\_and OPER\_LT expr
73. | expression\_and OPER\_GE expr
74. | expression\_and OPER\_LE expr
75. | expr
76. expr: expr OPER\_LEFT expr\_shift
77. | expr OPER\_RIGHT expr\_shift
78. | expr\_shift
79. expr\_shift: expr\_shift OPER\_PLUS term
80. | expr\_shift OPER\_MINUS term
81. | term
82. term: term '\*' factor
83. | term OPER\_DIV factor
84. | term OPER\_MOD factor
85. | factor
86. factor: OPER\_MINUS factor
87. | OPER\_NOT factor
88. | number
89. number: name '[' expression ']'
90. | name '.' name
91. | '(' expression ')'
92. | name '(' expression\_list ')'
93. | const\_value
94. | name
95. | '\*' name '[' expression ']'
96. | '\*' name '.' name
97. | '\*' name '(' expression\_list ')'
98. | '\*' name
99. | '\*' '(' expression ')'
100. | '&' name
101. | '&' name '[' expression ']'
102. expression\_list: expression\_list ',' expression
103. | expression
104. |
105. lambda\_expression: LAMBDA '(' para\_list ')' ':' var\_type OPER\_PTR compound\_statement
106. | LAMBDA '(' para\_list ')' ':' var\_type OPER\_PTR expression
107. | LAMBDA '(' para\_list ')' OPER\_PTR compound\_statement
108. | LAMBDA '(' para\_list ')' OPER\_PTR expression
109. assign\_statement
110. : name '=' expression
111. | name '=' lambda\_expression
112. | name '[' expression ']' '=' expression
113. name: IDENTIFIER
114. class\_declaration: CLASS name inherit\_part '{' class\_body '}'
115. inherit\_part: ':' name
116. |
117. class\_body: class\_init class\_del func\_declaration\_list
118. class\_ini: FUNCTION INIT '(' var\_list ')' OPER\_PTR var\_type compound\_statement
119. class\_del: FUNCTION DEL '(' var\_list ')' OPER\_PTR var\_type compound\_statement
120. func\_declaration\_list: func\_declaration\_list func\_declaration
121. |

## 不带行数版

program

: global\_area main\_func ;

global\_area

: global\_area const\_declaration ';'

| global\_area var\_declaration ';'

| global\_area func\_declaration

| global\_area class\_declaration

|

;

const\_declaration

: LET const\_list

;

const\_list

: const\_list ',' const\_expr

| const\_expr

;

const\_expr

: name '=' const\_value

;

const\_value

: INTEGER

| FLOAT

| DOUBLE

| CHAR

| BOOLEAN

| STRING

;

var\_declaration

: VAR var\_list

;

para\_list

:

| var\_list

;

var\_list

: var\_list ',' var\_expr

| var\_expr

;

var\_expr

: name '=' expression

| name '=' lambda\_expression

| name ':' type\_declaration

| name

;

type\_declaration

: var\_type

| array\_type\_declaration

| func\_type

;

array\_type\_declaration

: var\_type '[' INTEGER ']'

;

func\_type

: FUNCTION

;

var\_type

: TYPE\_INT

| TYPE\_INT\_POINTER

| TYPE\_INT\_64

| TYPE\_INT\_64\_POINTER

| TYPE\_CHAR

| TYPE\_CHAR\_POINTER

| TYPE\_FLOAT

| TYPE\_FLOAT\_POINTER

| TYPE\_DOUBLE

| TYPE\_DOUBLE\_POINTER

| TYPE\_BOOL

| TYPE\_BOOL\_POINTER

;

func\_declaration

: FUNCTION name '(' para\_list ')' compound\_statement

| FUNCTION name '(' para\_list ')' OPER\_PTR var\_type compound\_statement

;

main\_func

: FUNCTION MAIN '(' ')' OPER\_PTR TYPE\_INT compound\_statement

;

statement\_list

: statement\_list statement

|

;

statement

: compound\_statement

| branch\_statement

| for\_statement

| jump\_statement ';'

| assign\_statement ';'

| var\_declaration ';'

| const\_declaration ';'

| name '(' expression\_list ')' ';'

;

compound\_statement

: '{' statement\_list '}'

;

branch\_statement

: IF '(' expression ')' compound\_statement

| IF '(' expression ')' compound\_statement ELSE compound\_statement

;

for\_statement

: WHILE '(' expression ')' compound\_statement

| FOR name IN '[' expression ',' expression ',' expression ']' compound\_statement

;

jump\_statement

: JUMP\_BREAK

| JUMP\_CONTINUE

| JUMP\_RETURN

| JUMP\_RETURN expression

;

expression

: expression OPER\_OR expression\_or

| expression\_or

;

expression\_or

: expression\_or OPER\_AND expression\_and

| expression\_and

;

expression\_and

: expression\_and OPER\_EQ expr

| expression\_and OPER\_NE expr

| expression\_and OPER\_GT expr

| expression\_and OPER\_LT expr

| expression\_and OPER\_GE expr

| expression\_and OPER\_LE expr

| expr

;

expr

: expr OPER\_LEFT expr\_shift

| expr OPER\_RIGHT expr\_shift

| expr\_shift

;

expr\_shift

: expr\_shift OPER\_PLUS term

| expr\_shift OPER\_MINUS term

| term

;

term

: term '\*' factor

| term OPER\_DIV factor

| term OPER\_MOD factor

| factor

;

factor

: OPER\_MINUS factor

| OPER\_NOT factor

| number

;

number

: name '[' expression ']'

| name '.' name

| '(' expression ')'

| name '(' expression\_list ')'

| const\_value

| name

| '\*' name '[' expression ']'

| '\*' name '.' name

| '\*' name '(' expression\_list ')'

| '\*' name

| '\*' '(' expression ')'

| '&' name

| '&' name '[' expression ']'

;

expression\_list

: expression\_list ',' expression

| expression

|

;

lambda\_expression

: LAMBDA '(' para\_list ')' ':' var\_type OPER\_PTR compound\_statement

| LAMBDA '(' para\_list ')' ':' var\_type OPER\_PTR expression

| LAMBDA '(' para\_list ')' OPER\_PTR compound\_statement

| LAMBDA '(' para\_list ')' OPER\_PTR expression

;

assign\_statement

: name '=' expression

| name '=' lambda\_expression

| name '[' expression ']' '=' expression

;

name

: IDENTIFIER

;

class\_declaration

: CLASS name inherit\_part '{' class\_body '}'

;

inherit\_part

: ':' name

|

;

class\_body

: class\_init class\_del func\_declaration\_list

;

class\_init

: FUNCTION INIT '(' var\_list ')' OPER\_PTR var\_type compound\_statement

;

class\_del

: FUNCTION DEL '(' var\_list ')' OPER\_PTR var\_type compound\_statement

;

func\_declaration\_list

: func\_declaration\_list func\_declaration

|