

<加法运算符> ::= + | -

<乘法运算符> ::= * | /

<关系运算符> ::= < | <= | > | >= | != | ==

<字母> ::= _ | a | . . . | z | A | . . . | Z

<数字> ::= 0 | <非零数字>

<非零数字> ::= 1 | . . . | 9

<字符> ::= '<加法运算符>' | '<乘法运算符>' | '<字母>' | '<数字>'

<字符串> ::= " {十进制编码为 32,33,35-126 的 ASCII 字符} "

<程序> ::= [<常量说明>] [<变量说明>] { <有返回值函数定义> | <无返回值函数定义> } <主函数>

<常量说明> ::= const <常量定义>; { const <常量定义>; }

<常量定义> ::= int <标识符> = <整数> { , <标识符> = <整数> }

| char <标识符> = <字符> { , <标识符> = <字符> }

<无符号整数> ::= <非零数字> { <数字> } | 0

<整数> ::= [+ | -] <无符号整数>

<标识符> ::= <字母> { <字母> | <数字> }

<声明头部> ::= int <标识符> | char <标识符>

<变量说明> ::= <变量定义>; { <变量定义>; }

<变量定义> ::= <类型标识符> (<标识符> | <标识符> '[' <无符号整数> ']') { (<标识符> | <标识符> '[' <无符号整数> ']') }

// <无符号整数> 表示数组元素的个数，其值需大于 0

<类型标识符> ::= int | char

<有返回值函数定义> ::= <声明头部> '(' <参数表> ')' '{ <复合语句> }'

<无返回值函数定义> ::= void <标识符> '(' <参数表> ')' '{ <复合语句> }'

<复合语句> ::= [<常量说明>] [<变量说明>] <语句列>

<参数表> ::= <类型标识符> <标识符> { , <类型标识符> <标识符> } | <空>

<主函数> ::= void main '(' ')' '{ <复合语句> }'

<表达式> ::= [+ | -] <项> { <加法运算符> <项> } // [+ | -] 只作用于第一个 <项>

<项> ::= <因子> { <乘法运算符> <因子> }

<因子> ::= <标识符> | <标识符> '[' <表达式> ']' '(' <表达式> ')' | <整数> | <字符> | <有返回值函数调用语句>

<语句> ::= <条件语句> | <循环语句> | '{ <语句列> }' | <有返回值函数调用语句>;

| <无返回值函数调用语句>; | <赋值语句>; | <读语句>; | <写语句>; | <空>; | <返回语句>;

<赋值语句> ::= <标识符> = <表达式> | <标识符> '[' <表达式> ']' = <表达式>

<条件语句> ::= if '(' <条件> ')' <语句> [else <语句>]

<条件> ::= <表达式> <关系运算符> <表达式> // 整型表达式之间才能进行关系运算

| <表达式> // 表达式为整型，其值为 0 条件为假，值不为 0 时条件为真

<循环语句> ::= while '(' <条件> ')' <语句> | do <语句> while '(' <条件> ')' | for '(' <标识符> = <表达式>; <条件>; <标识符>

= <标识符> (+ | -) <步长> ')' <语句>

<步长> ::= <无符号整数>

<有返回值函数调用语句> ::= <标识符> '(' <值参数表>)'

<无返回值函数调用语句> ::= <标识符> '(' <值参数表>)'

<值参数表> ::= <表达式> { , <表达式> } | <空>

<语句列> ::= { <语句> }

<读语句> ::= scanf '(' <标识符> { , <标识符> })'

<写语句> ::= printf '(' <字符串> , <表达式> ')' | printf '(' <字符串> ')' | printf '(' <表达式>)'

<返回语句> ::= return '(' <表达式>)'