＜加法运算符＞ ::= +｜-

作用：进行加法运算

限定条件：只能为’+‘或者’-’

句子示例：a + b，a - b

＜乘法运算符＞ ::= \*｜/

作用：进行乘法或除法运算

限定条件：只能为‘\*’或者‘/’

句子示例：a \* b，a / b

＜关系运算符＞ ::= <｜<=｜>｜>=｜!=｜==

作用：进行比较关系运算

限定条件：只能为其中的一种

句子示例：a < b, a <= b, a > b, a != b, a == b

＜字母＞ ::= ＿｜a｜．．．｜z｜A｜．．．｜Z

作用：表示一个字母

限定条件：必须为其中的一种

句子示例：a

＜数字＞ ::= ０｜＜非零数字＞

作用：表示一个数字

限定条件：必须全部由数字组成

句子示例：2333

＜非零数字＞ ::= １｜．．．｜９

作用：表示一个非零数字

限定条件：必须为1到9之间的一个

句子示例：1

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

作用：表示一个字符

限定条件：只能是运算符或字母或数字，要用‘’括起来

句子示例：a

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

作用：规定一个字符串的组成

限定条件：必须全部由十进制编码为32,33,35-126的ASCII字符组成，要用””括起来

句子示例：#abc

＜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＞

作用：构成一个程序

限定条件：常量说明要在变量说明的前面，有没有都可以，非主函数要全部定义在主函数的前面，函数返回值可选

句子示例：见程序示例

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

作用：声明一个常量

限定条件：必须以const关键字开始，多个常量定义之间要以分号间隔

句子示例：const int a = 1; const char b = 'c';

＜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＞}

作用：定义一个常量表达式

限定条件：类型只能为int或char，多个同类型常量声明要用逗号隔开

句子示例：int a = 1, b = 2; char c = 'd', e = 'f';

＜无符号整数＞ ::= ＜非零数字＞｛＜数字＞｝

作用：表示一个无符号整数

限定条件：必须以非零数字开始

句子示例：2333

＜整数＞ ::= ［＋｜－］＜无符号整数＞｜０

作用：表示一个整数

限定条件：0之前不能加正负号

句子示例：+2333

＜标识符＞ ::= ＜字母＞｛＜字母＞｜＜数字＞｝

作用：表示一个标识符

限定条件：必须以字母开始，只能包含字母或者数字

句子示例：a123

＜声明头部＞ ::= int＜标识符＞|char＜标识符＞

作用：表示一个头部声明

限定条件：必须以int或char开始，紧跟一个标识符结束

句子示例：int a

＜变量说明＞ ::= ＜变量定义＞;{＜变量定义＞;}

作用：表示一个变量

限定条件：不同的变量定义之间要以分号隔开

句子示例：int a; char b;

＜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＞‘]’) }

作用：定义一个变量

限定条件：必须以一个类型标识符开始，接着至少包含一个标识符说明，多个标识符之间要用逗号隔开

句子示例：int a, b;

＜类型标识符＞ ::= int | char

作用：表示一个类型标识符

限定条件：类型只能为int或char

句子示例：int

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

作用：声明一个有返回值的函数

限定条件：必须以声明头部开始，接着在小括号里写参数，接着在花括号里写符合语句

句子示例：int foo(int a, char b) { }

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

作用：声明一个没有返回值的函数

限定条件：必须以void开始，紧跟一个标识符，接着在小括号里写参数，接着在花括号里写符合语句

句子示例：void foo(int a, char b) { }

＜复合语句＞ ::= ［＜常量说明＞］［＜变量说明＞］＜语句列＞

作用：表示一个复合语句

限定条件：常量说明以及变量说明必须要在语句列的前面，有没有可选

句子示例：const int a = 1; char b = 'c'; return;

＜参数＞ ::= ＜参数表＞

作用：表示一个参数

限定条件：必须由一个参数表组成

句子示例：int a, char b

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

作用：表示一个参数表

限定条件：为空或有一个以上的参数，参数声明以类型标识符开始，紧跟着标识符

句子示例：int a, char b

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

作用：表示一个主函数

限定条件：必须以void main()开始，紧跟着花括号里要有复合语句

句子示例：void main(){return;}

＜表达式＞ ::= ［＋｜－］＜项＞{＜加法运算符＞＜项＞}

作用：表示一个表达式

限定条件：可以用一个加号或减号开始，必须由两个项，两个项之间必须用加法运算符分隔

句子示例：a + b

＜项＞ ::= ＜因子＞{＜乘法运算符＞＜因子＞}

作用：表示一个项

限定条件：必须以一个因子开始，如果有多个因子，两个因子之间要用乘法运算符分隔

句子示例：a \* b

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

作用：表示一个因子

限定条件：必须为以上规定中的一种

句子示例：a(b[1+1])

＜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＞;

作用：表示一个语句

限定条件：必须为以上规定中的一种

句子示例：{return;}

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

作用：表示一个赋值语句

限定条件：必须只有一个等号，等号左边必须为一个标识符或标识符[表达式]，右边必须为一个表达式

句子示例：a = b + c;

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

作用：表示一个条件语句

限定条件：必须以if开头，紧接着一个小括号，括号里只能有一个条件，括号后面紧跟着一个语句，语句后面可以选择跟一个或多个else语句块

句子示例：if(a > 1) {return;} else {return;}

＜条件＞ ::= ＜表达式＞＜关系运算符＞＜表达式＞｜＜表达式＞ //表达式为0条件为假，否则为真

作用：表示一个条件

限定条件：必须以表达式开始，如果有多个表达式，表达式之间要用关系运算符隔开

句子示例：a > 1

＜循环语句＞ ::= 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＞

作用：表示一个循环语句

限定条件：如果用do...while结构，do和while之间有且只有一个语句，while后面紧跟一个小括号，括号里是条件，括号后跟着语句

如果用for循环，必须以for开头，后面紧跟小括号，括号里包括三项，第一项要以标识符开始，以表达式结束，中间以等号分隔，第二项有且只有一个

条件语句，第三项要以标识符开始，标识符后面跟着等号，再后面是另一个标识符，以步长结束，中间必须用加号或减号分隔

句子示例：do { } while(i > 0);for(i = 0; i < 10; i = i + 1) { }

＜步长＞::= ＜无符号整数＞

作用：表示一个步长

限定条件：必须为一个无符号整数

句子示例：1

＜有返回值函数调用语句＞ ::= ＜标识符＞‘(’＜值参数表＞‘)’

作用：表示一个用返回值函数的调用

限定条件：必须以标识符开始，后跟小括号，括号内为值参数表

句子示例：a(b)

＜无返回值函数调用语句＞ ::= ＜标识符＞‘(’＜值参数表＞‘)’

作用：表示一个没有返回值的函数调用语句

限定条件：必须以标识符开始，后跟小括号，括号内为值参数表

句子示例：a(b)

＜值参数表＞ ::= ＜表达式＞{,＜表达式＞}｜＜空＞

作用：表示一个值参数表

限定条件：可以为空，或者有一个及以上的表达式，表达式之间用逗号分隔

句子示例：a, b

＜语句列＞ ::=｛＜语句＞｝

作用：表示一个语句列

限定条件：必须有零个或以上的语句组成

句子示例：a = b; b = c;

＜读语句＞ ::= scanf ‘(’＜标识符＞{,＜标识符＞}‘)’

作用：表示一个读入语句

限定条件：必须以scanf开始，紧跟着小括号，括号里可以有一个或以上的标识符，不同的标识符之间用逗号分隔

句子示例：scanf(a)

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

作用：表示一个写出语句

限定条件：必须以printf开始，紧跟着一个小括号，括号里可以包含一个字符串或者一个表达式，或者一个字符串和表达式对

句子示例：printf("hello world");

＜返回语句＞ ::= return[‘(’＜表达式＞‘)’]

作用：表示一个返回语句

限定条件：必须以return开始，如果紧跟一个小括号，括号里必须是一个表达式

句子示例：return(0)