＜加法运算符＞ ::= +｜-

**解释：加法运算符由加法符号和减法符号组成。**

**限定：用于表达式的计算。**

**实例：a+b，6-c。**＜乘法运算符＞  ::= \*｜/

**解释：乘法运算符由乘法符号和除法符号组成。**

**限定：用于表达式的计算。**

**实例：4\*c，d/e。**  
＜关系运算符＞  ::=  <｜<=｜>｜>=｜!=｜==

**解释：定义关系运算符，有小于，大于，小于等于，大于等于，恒等于，不等于**

**限定：关系比较运算，结果只有0和1两种情况。**

**实例：a>=3,b==c.**  
＜字母＞   ::= ＿｜a｜．．．｜z｜A｜．．．｜Z

**解释：由字母以及下划线组成。**

**限定：用于标识符的定义，字符串的表示。**

**实例：a，r，\_。**  
＜数字＞   ::= ０｜＜非零数字＞

**解释：单个数字组成的集合。**

**限定：用于数的表示。**

**实例：0，1，6。**  
＜非零数字＞  ::= １｜．．．｜９

**解释：非零数字组成的集合。**

**限定：用于数的表示，用来确定数的首位不是0。**

**实例：1，5，3。**  
＜字符＞    ::=  '＜加法运算符＞'｜'＜乘法运算符＞'｜'＜字母＞'｜'＜数字＞'

**解释：定义单字符，由加法运算符、乘法运算符、字母、数字组成。**

**限定：单字符常量、标识符的定义，表示单位数字和运算符。**

**实例：“a”，“+”，“4”。**  
＜字符串＞   ::=  "｛十进制编码为32,33,35-126的ASCII字符｝"

**解释：定义字符串，由十进制编码为32,33,35-126的ASCII字符组成。**

**限定：字符串中的字符由十进制编码为32,33,35-126的ASCII字符组成。**

**实例：“dsads”，“543sa+as1”，“&3fs”。**  
＜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＞

**解释：构成了C0文法支持的代码的结构。**

**限定：程序自上向下的组成分别是：全局常量说明部分（可有可无）、全局变量说明部分（可有可无）、有返回值的函数定义或者无返回值的函数定义（可有可无）、主函数。**

**实例：const int c;**

**int s,p,q;**

**int ca(int t){**

**return 2\*t;**

**}**

**void main(){**

**s = c;**

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

**解释：定义常量说明部分的组成。**

**限定：声明必须以const开头，以;结尾。**

**实例：const int c = 14；。**  
＜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＞}

**解释：定义常量定义的组成。**

**限定：声明必须以int或者char开头，以;结尾。**

**实例：const int c = 14；const char s = ‘r’。**  
＜无符号整数＞  ::= ＜非零数字＞｛＜数字＞｝| 0

**解释：定义无符号整数的组成。**

**限定：除0外，无符号整数的最前位不能为0。**

**实例：325，0；。**  
＜整数＞        ::= ［＋｜－］＜无符号整数＞

**解释：定义所有整数的组成。**

**限定：加号和减号可以忽略。除0外，无符号整数的最前位不能为0。**

**实例：0，432，+452，-31；。**  
＜标识符＞    ::=  ＜字母＞｛＜字母＞｜＜数字＞｝

**解释：定义所有标识符的组成。**

**限定：标识符首位只能由字母或者下划线组成，其余部分只能由数字、字母或下划线组成。**

**实例：dsss,\_ddsw；。**

＜声明头部＞   ::=  int＜标识符＞ |char＜标识符＞

**解释：定义所有声明头部的组成，用于有返回值的函数定义。**

**限定：声明头部的最前部分只能说int或者char。**

**实例：int s（int c）{**

**return（c+1）}；。**  
＜变量说明＞  ::= ＜变量定义＞;{＜变量定义＞;}

**解释：定义变量说明部分的组成。**

**限定：全局变量声明。**

**实例：int c;char s;**

＜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

**解释：定义变量定义部分的组成。**

**限定：必须以类型标识符开头，一次可以声明多个同类型的变量。**

**实例：int c,char s[10];**  
＜常量＞   ::=  ＜整数＞|＜字符＞

**解释：定义变量部分的组成。**

**限定：常量只能是整数或者单字符。**

**实例：543，f;**  
＜类型标识符＞      ::=  int | char

**解释：定义类型标识符部分的组成。**

**限定：只能是int或者char。**

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

**解释：定义有返回值的函数定义的组成。**

**限定：必须有函数的声明头部，后面必须有括号，复合语句必须由在大括号里面。**

**实例：int cc(int r){return (2\*r);};**  
＜无返回值函数定义＞  ::= void＜标识符＞'('＜参数表＞')''{'＜复合语句＞'}'

**解释：定义无返回值的函数类型的组成。**

**限定：开头必须是void，后接标识符，后面必须有括号，复合语句必须在大括号里面。**

**实例：void dd(int c){c = c+1;(return);}**  
＜复合语句＞   ::=  ［＜常量说明＞］［＜变量说明＞］＜语句列＞

**解释：定义复合语句的组成。**

**限定：用在函数里面，开始时常量说明部分（可有可无），然后是变量说明部分（可有可无），后接语句列。**

**实例：const int c = 2;int e;e = c;;**  
＜参数表＞    ::= ＜参数＞{,＜参数＞}| ＜空>

**解释：定义参数表的组成。**

**限定：参数表可以为空，也可以有多个参数，用在函数定义的参数里。**

**实例：int t，char s;**  
＜参数＞    ::=  ＜类型标识符＞＜标识符＞

**解释：定义参数的组成。**

**限定：参数的开头只能为int 或者char ，后接标识符。**

**实例：int c，char t;**  
＜主函数＞    ::= void main'('')''{'＜复合语句＞'}'

**解释：定义主函数部分的组成。**

**限定：只能以void main()开头,后面必须接大括号。**

**实例：void main(){return ;};**  
＜表达式＞    ::= ［＋｜－］＜项＞{＜加法运算符＞＜项＞}   //[+|-]只作用于第一个<项>

**解释：定义表达式部分的组成。**

**限定：[+|-]只作用于第一个<项>，用于求值过程。**

**实例：a+c-8;**  
＜项＞     ::= ＜因子＞{＜乘法运算符＞＜因子＞}

**解释：定义项部分的组成。**

**限定：开头必须是因子，组成表达式。**

**实例：3\*5/cd;**  
＜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＞

**解释：定义因子部分的组成。**

**限定：因子可以是变量、数组某个元素、表达式的结果、整数、字符或者有返回值的函数调用语句。**

**实例：cd，a[4],‘c’，3，return (0),c+w;**       
＜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＞;|＜情况语句＞｜＜返回语句＞;

**解释：定义语句部分的组成。**

**限定：语句可以是条件语句、循环语句、带大括号的语句列、有返回值的函数调用语句、无返回值的函数调用语句、赋值语句、读语句、写语句、空、情况语句、返回语句，大多数情况下后接;。**

**实例：if(c==3){**

**While(c){**

**c--;**

**}**

**}**

**Printf(“c = 0”);**  
＜赋值语句＞   ::=  ＜标识符＞＝＜表达式＞|＜标识符＞'['＜表达式＞']'=＜表达式＞

**解释：定义赋值语句部分的组成。**

**限定：可以赋值给某个变量，也可以赋值给某个数组元素。**

**实例：c = a；a[2] = g;**  
＜条件语句＞  ::=  if '('＜条件＞')'＜语句＞

**解释：定义条件语句部分的组成。**

**限定：必须以if开头，后接括号。**

**实例：if(c>=0) c = -1;**  
＜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

**解释：定义条件部分的组成。**

**限定：结果只能为0或者1。**

**实例：a<3 c;**  
＜循环语句＞   ::=  while '('＜条件＞')'＜语句＞

**解释：定义部循环语句部分的组成。**

**限定：必须以while开头，后接括号，与条件语句类似。**

**实例：while(c>0) c = c-1;**  
＜情况语句＞  ::=  switch '('＜表达式＞')' '{'＜情况表＞＜缺省＞ '}'

**解释：定义情况语句部分的组成。**

**限定：必须以switch部分开头，后接小括号，后面包括情况表和缺省。**

**实例：switch(c){**

**case 10:c = c-1;**

**case 9:c = c-2;**

**default:printf(“ok”);**

＜情况表＞   ::=  ＜情况子语句＞{＜情况子语句＞}

**解释：定义情况表部分的组成。**

**限定：只由情况子语句组成。**

**实例：case 10：c = 4;**

**case 7:d = 4;**  
＜情况子语句＞  ::=  case＜常量＞：＜语句＞

**解释：定义情况子语句部分的组成。**

**限定：必须以case开头，后接常量和冒号，后接语句。**

**实例：同上;**  
＜缺省＞   ::=  default : ＜语句＞|＜空＞

**解释：定义缺省部分的组成。**

**限定：必须以default开头，后接冒号，后接语句或者不接。**

**实例：default:printf(“ok”);**  
＜有返回值函数调用语句＞ ::= ＜标识符＞'('＜值参数表＞')'

**解释：定义有返回值函数调用语句部分的组成。**

**限定：必须以标识符开头，后接括号，括号里为值参数表。**

**实例：b = sss(3,2);**  
＜无返回值函数调用语句＞ ::= ＜标识符＞'('＜值参数表＞')'

**解释：定义无返回值函数调用语句部分的组成。**

**限定：必须以标识符开头，后接括号，括号里为值参数表。**

**实例：sss(3,2);**  
＜值参数表＞   ::= ＜表达式＞{,＜表达式＞}｜＜空＞

**解释：定义值参数表部分的组成。**

**限定：后接0个或者多个表达式。**

**实例：33，d+g;**  
＜语句列＞   ::= ｛＜语句＞｝

**解释：定义语句列部分的组成。**

**限定：可以由多条语句组成。**

**实例：c = c+1；d = d+1;**  
＜读语句＞    ::=  scanf '('＜标识符＞{,＜标识符＞}')'

**解释：定义读语句部分的组成。**

**限定：必须以scanf开头，后接括号，括号里为一个或者多个标识符。**

**实例：scanf(s,d);**  
＜写语句＞    ::= printf '(' ＜字符串＞,＜表达式＞ ')'| printf '('＜字符串＞ ')'| printf '('＜表达式＞')'

**解释：定义写语句部分的组成。**

**限定：必须以printf开头，后接括号，括号里为字符串或者表达式。**

**实例：printf(“ddd:”,d);**  
＜返回语句＞   ::=  return['('＜表达式＞')']

**解释：定义返回语句部分的组成。**

**限定：必须以return开头，后接括号，括号里为一个表达式(可有可无)。**

**实例：return(s,d); return;**