1.<加法运算符> ::= +｜-

定义加减法运算符

例子：3+4 a+b 3+a

2.<乘法运算符> ::= \*｜/

定义乘除法运算符

例子：1/2 a/2 a/b

3.<关系运算符> ::= <｜<=｜>｜>=｜!=｜==

定义关系运算符

例子：1<2 a>=3 b!=8

4.<逻辑运算符> ::= && | '|| ' | !

5.<字母> ::= ＿｜a｜．．．｜z｜A｜．．．｜Z

定义字母可以为大小写字母或者下划线

例子：\_ a B

6.<非零数字> ::= 1｜．．．｜9

定义非零数字为1~9的单位数字字符

例子： 1 8

7.<数字> ::= 0｜<非零数字>

定义数字为0~9的单位数字字符

例子： 0 1 8

8.<无符号整数> ::= <非零数字>｛<数字>｝

定义无符号整数的开头必为非零数字，其后可跟若干个数字

该限定表明以0开头的数字串不属于无符号整数

0就不是无符号整数

例子： 1 99

9.<整数> ::= ［＋｜－］<无符号整数>｜0

定义整数是以可省略的正负号开头，后跟无符号整数或者0的字符串

例子： 0 1 3

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

定义字符为以单引号开始和结束，其间有且仅有一个字符。

例子： '+' '/' 'a' '8'

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

定义字符串可以为0个，1个，或多个十进制编码为32,33,35-126的ASCII字符

需要注意的是不包括双引号,但是包括空格

字符串的第一个和最后一个字符为双引号

例子："hello, world~"

12.<标识符> ::= <字母>｛<字母>｜<数字>｝

定义标志符是必须由字母为开头，后跟0到多个字母或者数字的字符串

13.<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>

定义程序。

限定程序的子成分顺序，必须为常量说明在前，然后是变量说明，以上两者均可没有。

然后是有返回或者无返回的函数定义若干（可以是0个），然后才是主函数

例子: const int a = 3;

int b = 4;

void f(int a);

int f2();

void main(){...}

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

定义常量说明可以是一个或者多个const<常量定义>；这样的格式

例子： const int a = 2;

const char c = 'b';

15.<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>}

定义常量定义可以是整型常量或者字符型常量

整型常量的定义格式是int后跟一个或多个<标识符>＝<整数>的形式，之间用逗号隔

字符型常量同理

例子： int a = 2;

char b = 'b';

16.<变量说明> ::= <变量定义>;{<变量定义>;}

定义变量说明为一个或多个变量定义和分号组成的字符串

17.<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>‘]’ )}

定义变量定义，变量可以为一个标志符或者由标志符为起始的数组形式。

例子： int a2bc\_d

int word[10]

但是不能如下定义：

int 34dfn

int word[-1]

18.<声明头部> ::= int<标识符> | char<标识符>

定义声明头部

19.<类型标识符> ::= int | char

定义类型标识符为整数和字符这两种

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

定义有返回值函数。必须包含头声明头部，参数和复合语句以及必要的括号

例子： int f(char a){

...

...

}

char f(char a){

...

...

}

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

没啥好说的。

例子： void f(char a){

...

...

}

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

参数表由若干个类型标识符和标志符的集合组成，其间用逗号隔开。可以为空。

类型标识符和标志符的顺序不能反。

需要注意的事不能有数组形式

例如： int a, char b, int c,

23.<复合语句> ::= ［<常量说明>］［<变量说明>］<语句列>

定义复合语句是现场两再变量再语句的顺序。

例子： const int a = 2;

char b = '4';

...

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

例子： void main() {

...

}

25.<算术表达式> ::= ［＋｜－］<项>{<加法运算符><项>}

表达式是由可省略的+-开头的若干个由加法运算符连接的项的字符串

例如： 3\*4+2/6-4\*a

26.<项> ::= <因子>{<乘法运算符><因子>}

项是由乘法运算符连接的一个或多个因子

例如： 4\*b\*a

13\*6\*num[15]

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

因子是由标志符或标志符后跟方括号括起来的表达式或整数或者字符或者有返回值得函数调用语句或以圆括号括起来的表达式

例如： num[15]

num[a]

a

4

(3\*13+5\*a-2/f)

~~31.<布尔表达式> ::= <算术表达式><关系运算符><算术表达式>｜<算术表达式>~~

~~表达式为0条件为假，否则为真，值得注意的是一个条件中只有一个关系运算符，不支持嵌套~~

~~例如：~~

~~1&2 a&b a 0 1~~

~~31. <布尔表达式> ::= <布尔表达式> ‘||’ <布尔项> | <布尔项>~~

~~32.<布尔项> ::= <布尔项> && <布因子> | <布因子>~~

~~33.<布因子> ::= ! <布因子> | ‘(‘<布尔表达式>’)’ | true | false | <算术表达式> <关系运算符> <算术表达式>~~

文法改造：（消除左递归）

28.<布尔表达式> ::= <布尔项> <布尔表达式>’

29.<布尔表达式>’ ::= ‘||’ <布尔项> <布尔表达式>’

30.<布尔表达式>’ ::= <空语句>

31.<布尔项> ::= <布因子><布尔项>’

32.<布尔项>‘ ::= && <布因子><布尔项>’

33.<布尔项> ::= <空语句>

34.<布因子> ::= <布尔量>

35.<布因子> ::= ! <布因子>

36.<布尔量> ::= false | true

37.<布尔量> ::= ‘(‘<布尔表达式>’)’

38.<布尔量> ::= <条件因子> <条件运算符><条件因子>

39.<条件因子> ::= <标识符>｜<标识符>‘[’<算术表达式>‘]’｜<整数>|<字符>｜<有返回值函数调用语句>

事实上，这里<条件因子>不像（算术）表达式中一样，这里不支持嵌套，不支持’()’，因为布尔表达式设计起来符合嵌套，主要是括号不好处理；暂时这么设计，然后如果真要表达式的话，可以在前面先计算赋值给一个变量

40.<语句> ::= <条件语句>｜<循环语句>｜<情况语句>｜‘{’<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>;

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

例如： a[3] = 4

b = 5

42.<条件语句> ::= if ‘(’<布尔表达式>‘)’<语句>［else<语句>］

例如： if(a > b){

a = 1;

b = 2;

}

43.<循环语句> ::= while ‘(’<布尔表达式>‘)’<语句>| for‘(’<标识符>＝<算术表达式>;<布尔表达式>;<标识符>＝<标识符>(+|-)<步长>‘)’<语句>

循环语句具有如下形式：

while(...){

...

}

for(...;...;...){

...

}

其中大括号有时可省略

for的括号中各类型元素不能省略也不能乱位。

44.<步长> ::= <非零数字>｛<数字>}

步长为以非零数字开始数字串

形式和无符号整数类似

~~45.＜情况语句＞ ::= switch ‘(’＜表达式＞‘)’ ‘{’＜情况表＞ ‘}’~~

~~46.＜情况表＞ ::= ＜情况子语句＞{＜情况子语句＞}~~

~~47.＜情况子语句＞ ::= case ＜整数＞|＜字符＞：＜语句＞~~

45.<有返回值函数调用语句> ::= <标识符>‘(’<值参数表>‘)’

例如： f(2, 3)

46.<无返回值函数调用语句> ::= <标识符>‘(’<值参数表>‘)’

同上

47.<值参数表> ::= <算术表达式>{,<算术表达式>}｜<空>

值参数表是若干个（包括0）表达式的集合

48.<语句列> ::= ｛<语句>｝

语句列是若干（包括0）个连续语句的集合

49.<读语句> ::= scanf ‘(’<标识符>{,<标识符>}‘)’

定义读语句是以scanf起始，后接圆括号阔起来的一个或多个以逗号隔开的标识符。

例子: scanf(a, b)

50.<写语句> ::= printf ‘(’<字符串>‘)’| printf ‘(’<算术表达式>‘)’

printf(a)

printf("Hello, compilier!\n")

51.<返回语句> ::= return[‘(’<算术表达式>‘)’]

返回语句是以return开始，后接可有可无的以圆括号包围的表达式

例子： return 0