**自定义的文法，难度等同于高级C0**

|  |  |
| --- | --- |
| 文法 | 属性 |
| **基本** | |
| ＜加法运算符＞ ::= +｜- | 定义加减法运算符 |
| ＜乘法运算符＞ ::= \*｜/ | 定义乘除法运算符 |
| ＜关系运算符＞ ::= <｜<=｜>｜>=｜!=｜== | 定义关系运算符 |
| ＜字母＞ ::= ＿｜a｜．．．｜z｜A｜．．．｜Z | 定义字母可以为大小写字母或者下划线 |
| ＜数字＞ ::= 自然数 | 定义数字为自然数 |
| ＜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。 |
| ＜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 |
| ＜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。 |
| ＜常量说明＞ ::= const＜常量定义＞;{ const＜常量定义＞;} | 定义常量，之后不能修改 |
| ＜常量定义＞ ::= int＜标识符＞＝＜整数＞{,＜标识符＞＝＜整数＞};（目前无，） | 定义常量是整型常量 |
| ＜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 |
| ＜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 |
| ＜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 |
| ＜关键字＞ ::= C语言关键字 |  |
| **变量声明** | |
| ＜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 |
| ＜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。 |
| ＜类型标识符＞ ::= **int** | 定义类型标识符为整数 |
| **函数** | |
| ＜声明头部＞ ::= **int**＜标识符＞ | 定义声明头部 |
| ＜有返回值函数定义＞ ::= ＜声明头部＞‘(’＜参数＞‘)’ ‘{’＜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 |
| ＜无返回值函数定义＞ ::= void＜标识符＞‘(’＜参数＞‘)’ ‘{’＜复合语句＞‘}’ | 定义无返回值函数。 |
| ＜参数＞ ::= ＜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。 |
| ＜主函数＞ ::= void main ‘(’ ‘)’ ‘{’＜复合语句＞‘}’ | 定义主函数 |
| ＜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＞ |  |
| **表达式** | |
| ＜表达式＞ ::= ＜项＞{＜加法运算符＞＜项＞} | 定义表达式 |
| ＜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 |
| ＜因子＞ ::= ＜标识符＞｜＜标识符＞‘[’＜表达式＞‘]’｜＜整数＞|＜有返回值函数调用语句＞|‘(’＜表达式＞‘)’ | 因子是由标志符或数组或整数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 |
| **语句** | |
| ＜复合语句＞ ::= {＜常量说明＞|＜变量说明＞|＜语句列＞} | 定义复合语句为常量说明、变量说明、语句列的组合 |
| ＜语句＞ ::= ＜条件语句＞｜＜循环语句＞｜‘{’＜语句列＞‘}’｜＜读语句＞;｜＜写语句＞;｜＜有返回值函数调用语句＞;|＜无返回值函数调用语句＞;｜＜赋值语句＞;｜＜空＞;｜＜返回语句＞; | 定义语句 |
| ＜语句列＞ ::= ｛＜语句＞｝ |  |
| ＜读语句＞ ::= **scanf** ‘(’＜标识符＞‘)’ |  |
| ＜写语句＞ ::=  **printf** ‘(’＜字符串＞‘)’| **printf** ‘(’＜表达式＞‘)’ |  |
| ＜返回语句＞ ::= return[‘(’＜表达式＞‘)’] |  |
| ＜赋值语句＞ ::= ＜标识符＞＝＜表达式＞|＜标识符＞‘[’＜表达式＞‘]’=＜表达式＞ | 定义赋值语句 |
| ＜条件语句＞ ::= if ‘(’＜条件＞‘)’＜语句＞［else＜语句＞］ | 定义条件语句 |
| ＜条件＞ ::= ＜表达式＞＜关系运算符＞＜表达式＞ |  |
| ＜循环语句＞ ::= while ‘(’＜条件＞‘)’＜语句＞| for‘(’＜标识符＞＝＜表达式＞;＜条件＞;＜赋值语句＞‘)’＜语句＞ |  |