|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 错误 | ERROR\_TOKEN | 1 | { | LP | 2 |
| } | RP | 3 | PML | [ | 4 |
| PMR | ] | 5 | ; | SEMI | 6 |
| , | COMMA | 7 | int | INT\_CONST | 8 |
| float | FLOAT\_CONST | 9 | double | DOUBLE\_CONST | 10 |
| long | LONG\_CONST | 11 | char | CHAR\_CONST | 12 |
| void | VOID | 13 | return | RETURN | 14 |
| if | IF | 15 | else | ELSE | 16 |
| while | WHILE | 17 | for | FOR | 18 |
| break | BREAK | 19 | continue | CONTINUE | 20 |
| + | ADD | 21 | - | REDUCE | 22 |
| \* | MUTIPLY | 23 | / | DIVIDE | 24 |
| ++ | ADDSELF | 25 | -- | REDUCESELF | 26 |
| = | ASSIGN | 27 | == | EQ | 28 |
| += | ADD\_EQ | 29 | -= | REDUCE\_EQ | 30 |
| \*= | MUTIPLY\_EQ | 31 | /= | DIVIDE\_EQ | 32 |
| 数字 | NUM | 33 | 字符串常量 | STRING\_CONST | 34 |
| ‘ | SQUTOE\_L | 35 | ’ | SQUTOE\_R | 36 |
| “ | DQUTOE\_L | 37 | ” | DQUTOE\_R | 38 |
| int | INT | 39 | double | DOUBLE | 40 |
| char | CHAR | 41 | long | LONG | 42 |
| < | LESS | 43 | > | GREATER | 44 |
| % | MOD | 45 | != | NOTEQ | 46 |
| ( | BRACKETL | 47 | ) | BRACKETR | 48 |
| && | AND | 49 | || | OR | 50 |
| include | INCLUDE | 51 | 其他 | OTHERS | 52 |
| \0 | ENDFILE | 53 | 注释 | COMMENT | 54 |
| 表示符 | ID | 54 | float | FLOAT | 56 |
| const | CONST | 57 | main | MAIN | 58 |

‘‘’‘’‘’‘’“”

1.program → declaration-list

2.declaration-list → declaration-list declaration | declaration

3.declaration → var-declaration | fun-declaration

程序由声明的列表(或序列)组成，声明可以是函数或变量声明，顺序是任意的。至少必须有 一个声明。接下来是语义限制(这些在C中不会出现)。所有的变量和函数在使用前必须声明 (这避免了向后backpatching引用)。程序中最后的声明必须是一个函数声明，名字为main。 注意，C1缺乏原型，因此声明和定义之间没有区别(像C一样)。

4.var-declaration → type-specifier ID  | type-specifier ID[NUM];

5.type-specifier → int | void | float 变量声明或者声明了简单的整数或浮点类型变量，或者是基类型为整数或浮点的数组变量， 索引范围从0到NUM-1。注意，在C1中仅有的基本类型是整型和空类型。在一个变量声明中， 只能使用类型指示符int。void用于函数声明(参见下面)。也要注意，每个声明只能声明一 个变量。

6.fun-declaration → type-specifier ID(params) compound-stmt

7.params → param-list | void

8.param-list → param-list , param | param

9.param → type-specifier ID |type-specifier ID[]  函数声明由返回类型指示符、标识符以及在圆括号内的用逗号分开的参数列表组成，后面跟 着一个复合语句，是函数的代码。如果函数的返回类型是void，那么函数不返回任何值(即 是一个过程)。函数的参数可以是void(即没有参数)，或者一列描述函数的参数。参数后面 跟着方括号是数组参数，其大小是可变的。简单的整型参数由值传递。数组参数由引用来传 递(也就是指针)，在调用时必须通过数组变量来匹配。注意，类型“函数”没有参数。一个 函数参数的作用域等于函数声明的复合语句，函数的每次请求都有一个独立的参数集。函数 可以是递归的(对于使用声明允许的范围)。

10.compound-stmt → { local-declarations statement-list }   复合语句由用花括号围起来的一组声明和语句组成。复合语句通过用给定的顺序执行语句序 列来执行。局部声明的作用域等于复合语句的语句列表，并代替任何全局声明。

11.local-declarations → local-declarations var-declaration | empty

12.statement-list → statement-list statement | empty

注意声明和语句列表都可以是空的(非终结符empty表示空字符串，有时写作ε。)

13.statement → expression-stmt  | compound-stmt  | selection-stmt  | iteration-stmt  | return-stmt

14.expression-stmt → expression; |

表达式语句有一个可选的且后面跟着分号的表达式。这样的表达式通常求出它们一方的结 果。因此，这个语句用于赋值和函数调用。

15.selection-stmt → if (expression) statement  | if (expression) statement else statement

if语句有通常的语义：表达式进行计算；非0值引起第一条语句的执行；0值引起第二条语句的执行，如果它存在的话。这个规则导致了典型的悬挂else二义性，可以用一种标准的方法解决：else部分通常作为当前if的一个子结构立即分析(“最近嵌套”非二义性规则)。

16.iteration-stmt → while (expression) statement

while语句是C－中唯一的重复语句。它重复执行表达式，并且如果表达式的求值为非0，则 执行语句，当表达式的值为0时结束。

17.return-stmt → return; | return expression  返回语句可以返回一个值也可无值返回。函数没有说明为void就必须返回一个值。函数声明

为void就没有返回值。return引起控制返回调用者(如果它在main中，则程序结束)。

18.expression → var=expression | simple-expression

19.var→ ID | ID[expression]

表达式是一个变量引用，后面跟着赋值符号(等号)和一个表达式，或者就是一个简单的表达 式。赋值有通常的存储语义：找到由var表示的变量的地址，然后由赋值符右边的子表达式 进行求值，子表达式的值存储到给定的地址。这个值也作为整个表达式的值返回。var是简 单的(整型)变量或下标数组变量。负的下标将引起程序停止(与C不同)。然而，不进行下标 越界检查。

var表示C1比C的进一步限制。在C中赋值的目标必须是左值(l-value)，左值是可以由许多操

作获得的地址。在C1中唯一的左值是由var语法给定的，因此这个种类按照句法进行检查， 代替像C中那样的类型检查。故在C1中指针运算是禁止的。

20.simple-expression → additive-expression relop additive-expression  |additive-expression

21.relop → <= | < | > | >= | == | != | && | ||

简单表达式由无结合的关系操作符组成(即无括号的表达式仅有一个关系操作符)。简单表达 式在它不包含关系操作符时，其值是加法表达式的值，或者如果关系算式求值为ture，其值 为1，求值为false时值为0。

22.additive-expression → additive-expression addop term | term

 23.addop → + | -

24.term → term mulop factor | factor

25.mulop → \* | /

加法表达式和项表示了算术操作符的结合性和优先级。/符号表示整数除；即任何余数都被 截去。

26.factor → (expression) | var | call | NUM

因子是围在括号内的表达式；或一个变量，求出其变量的值；或者一个函数调用，求出函数 的返回值；或者一个NUM，其值由扫描器计算。数组变量必须是下标变量，除非表达式由单个ID组成，并且以数组为参数在函数调用中使用(如下所示)。

27.call → ID(args)

28.args → arg-list | empty

29.arg-list → arg-list,expression | expression

函数调用的组成是一个ID(函数名)，后面是用括号围起来的参数。参数或者为空，或者由逗 号分割的表达式列表组成，表示在一次调用期间分配的参数的值。函数在调用之前必须声明， 声明中参数的数目必须等于调用中参数的数目。函数声明中的数组参数必须和一个表达式匹 配，这个表达式由一个标识符组成表示一个数组变量。

————————————————

版权声明：本文为CSDN博主「猫山王」的原创文章，遵循 CC 4.0 BY-SA 版权协议，转载请附上原文出处链接及本声明。

原文链接：https://blog.csdn.net/PINBODEXIAOZHU/article/details/25394417