# CMM 解释器构造

组号： 20 组长： 颜聪

CMM（C Minus Minus/C- -）语言为C语言的一个子集，其基础定义如下：

语言结构：顺序结构（赋值语句、输入、输出）、选择语句（if-else）、循环结构（while）。这些语句结构和C语言的结构一样，允许嵌套。

表达式局限于关系表达式和算术表达式，运算的优先级为：算术运算、关系运算，并服从左结合规则。

算术表达式包括整数和实数上的运算、变量及“（）”、“”、“+”、“-”、“/”，运算符的优先级顺序为：“（）”大于“”和“/”大于“+”和“-”。

关系运算符包括：“<”、“==”、“<>”。

一条语句以“；”结束；程序由一条语句或者由“{”和“}”嵌套表达的复合语句。

支持多行注释（使用“/”和“ ”）。

支持数组运算，数组的下标必须是正整数，使用“[”和“]”表示数组下标。

变量的使用之前需要先声明，声明的方式和C语言一样。

十进制的整数与实数。

标识符（由数字、字母和下划线组成的串，但必须以字母或下划线开头的串）。

CMM 的词法规则

（1） 关键字： 见下表

（2） 专用符号：  +  -  \*  /  <  <=  >  >=  ==  !=  = ++ +=  -- -= \*= /=  ;  ,  (  )  [  ]  {  }  /\* \*/ //

（3） ID 正则表达式定义：

ID = [ \_ | letter]^+ [ letter | \_ |Digit]\*

Letter = a|..|z|A|..|Z

Digit = 0|..|9

1. 注释由 /\*...\*/ 围起来。

CMM 编码表：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 保留字 | 编码 | 记忆符 | 保留字 | 编码 | 记忆符 |
| If | 1 | IF | // | 24 | SCOM |
| else | 2 | ELSE | [ | 25 | LBRACKET |
| while | 3 | WHILE | ] | 26 | RBRACKET |
| read | 4 | READ | <= | 27 | LET |
| write | 5 | WRITE | > | 28 | GT |
| int | 6 | INT | >= | 29 | GET |
| double | 7 | DOUBLE | identifier | 30 | ID |
| string | 8 | STRING | literal\_int | 31 | LITERAL\_INT |
| + | 9 | PLUS | literal\_double | 32 | LITERAL\_DOUBLE |
| - | 10 | MINUS | literal\_string | 33 | LITERAL\_STRING |
| \* | 11 | MUL | 逻辑表达式 | 34 | LOGIC\_EXP |
| / | 12 | DIV | 多项式 | 35 | ADDTIVE\_EXP |
| = | 13 | ASSIGN | 项 | 36 | TERM\_EXP |
| < | 14 | LT | String项 | 37 | STRING\_EXP |
| == | 15 | EQ | return | 38 | RETURN |
| <> | 16 | NEQ | void | 39 | VOID |
| ( | 17 | LPARENT | ++ | 40 | PLUSPLUS |
| ) | 18 | RPARENT | += | 41 | PLUSEQUAL |
| ; | 19 | SEMI | -- | 42 | MINUSMINUS |
| { | 20 | LBRACE | -= | 43 | MINUSEQUAL |
| } | 21 | RBARCE | \*= | 44 | MULTIEQUAL |
| /\* | 22 | LCOM | /= | 45 | DIVEEQUAL |
| \*/ | 23 | RCOM |  |  |  |

CMM 的 BNF 语法如下：

1. program -> declaration\_list
2. declaration\_list -> declaration\_list declaration  |  declaration
3. declaration\_list -> declaration\_list declaration  |  declaration
4. var\_declaration -> type\_specifier ID  |  type\_specifier ID [ NUM ]

5. type\_ specifier -> INT | VOID

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 [ ]

10. compound\_ stmt -> { local\_ declarations statement\_ list }

11. local\_ declarations -> 1ocal\_ declarations var\_ declaration | empty

12. statement\_ list -> statement\_ list statement | empty

13. statement -> expression\_ stmt | compound\_ stmt | se1ection\_ stmt |iteration\_ stmt | return\_ stmt

14. expression\_ stmt -> expression ;

15. selection\_ stmt -> IF ( expression ) statement l IF ( expression) statement ELSE statement

16. iteration\_ stmt -> WHILE ( expression ) statement

17. return\_ stmt > RETURN | RETURN expression

18. expression -> var= expression | simple\_ expression

19. var-> ID | ID [ expression ]

20. simple\_ expression-> additive\_expression relop additive\_expression | additive\_ expression

21. re1op > LE | LT l GT l GE | EQ | NE

22. additive\_ expression -> additive\_ expression addop term | term

23. addop -> + | -

24. term -> term mulop factor | factor

25. mulop -> \* | /

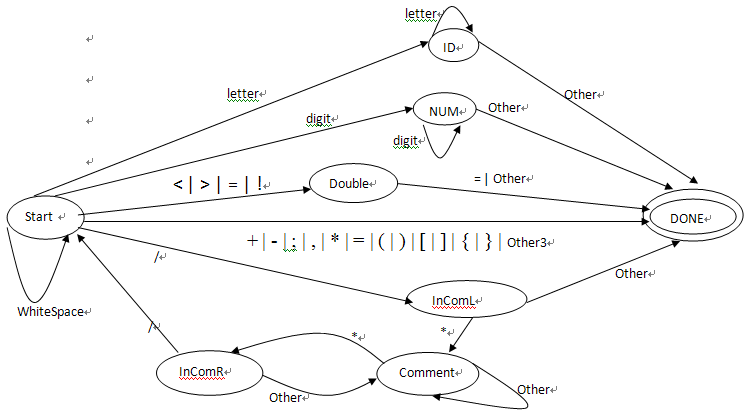
26. factor -> ( expression ) | var | call l NUM

27. call ->ID(args)

28. args -> arg\_ 1ist | empty

29. arg\_list > arg \_list , expression | expression

CMM DFA图



CMM 词法分析器主要流程：

