1. **词法规则**
2. **语言语法规则**

**CMM语言语法规则：**

1. 程序

<程序> ::= begin <语句串> end

2.语句串

<语句串> ::= {<语句>}

3.语句

<语句> ::= <声明语句>; | <赋值语句>; | <if-else语句> | <while-do语句>

4.声明语句

<声明语句> ::= (int | float | double) (标识符 | 标识符[整数])

5.赋值语句(拓展算术表达式到其他表达式)

<赋值语句> ::= (标识符 | 标识符[整数]) = <算术表达式>

6.if-else 语句

<if-else语句> ::= if (<逻辑表达式>) then { <语句> } else{<语句>}

7.while-do语句

<while-do语句> ::= while (<逻辑表达式>) do { <语句> }

7.逻辑表达式-----优先级问题 and or

<逻辑表达式> ::= <关系表达式>{||<关系表达式> | &&<关系表达式>}

8.关系表达式

<关系表达式> ::= <项>{< <项> | <=<项> | ><项> | >=<项> | <><项> | ==<项>

<项> ::= （<逻辑表达式>） | 标识符 | 数

8.算术表达式

<算术表达式> ::= <项>{+<项> | -<项>}

<项> ::= <因子>{\*<因子> | /<因子>}

<因子> ::= 标识符 | 标识符[整数] | 数 | （<算术表达式>）

9.标识符

标识符 ::= 字母{(字母|数字|下划线)}(字母 | 数字)

10.整数

整数 ::= 数字{数字} | -数字{数字}

11.小数

小数 ::= 数字{数字}.{数字} | -数字{数字}.{数字} | 数字{数字}.{数字}f | -数字{数字}.{数字}f

1. **文法定义**

**1.程序**

P 🡪 begin Z end

**2.语句串**

Z 🡪 SZ | ε

**3.语句**

S 🡪 A; | K; | I | W

**4.声明语句**

A 🡪 (int | float | double) IDB

B 🡪 [U] | ε

**4.赋值语句**

K 🡪 IDB = E

**5.if-then语句**

I 🡪 if (L) then {Z} else {Z}

**6.while-do语句**

W 🡪 while (L) do {Z}

**7.逻辑表达式**

L 🡪 RJ

J 🡪 || L | && L | ε

**8.关系表达式**

R 🡪 MQ

Q 🡪 == M | <> M | >= M | <= M | > M | < M |ε

(M change to R ?)

**9.算术表达式**

M 🡪 EV

V 🡪 + M | - M | ε

E 🡪 FT

T 🡪 \* E | / E | ε

F 🡪 (M) | IDB | N

**9.标识符**

ID 🡪 lDG

D 🡪 (l | d | \_)

G 🡪 (l | d)

**10.无符号整数**

N 🡪 dN | d

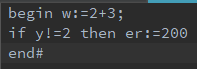
（\* 其中 l 代表字母， d代表数字）

1. **出错处理出口**

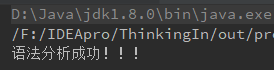
在识别换行符的过程中，设置了行号的标志整数，遇到换行符’\n’时会自动自增。这样一来，在发现错误时可以及时找到出错位置，即出错行，并反馈到输出文件上。当某程序在识别语言时，不断地在调用子程序，任何子程序在发现错误时，就会报出该子程序对应的错误，并显示在输出文件上。例如在表达式程序中出现了只有左括号没有右括号的情况，则在其某一子程序中会显示“括号不匹配，缺少右括号的”的错误提示。遇到错误时，程序对语言的语法分析会对于该错误通过错误信息的方式，包装一个Exception并返回，打印到屏幕上。

1. **测试计划**

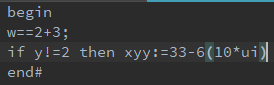
1．

**输入**：

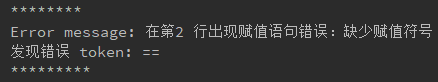
**输出**：



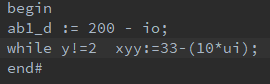
2．（赋值语句出错）

**输入**：

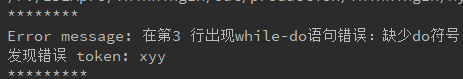
**输出**：



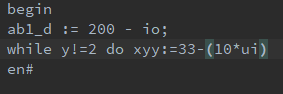
3．（while-do语句出错）

**输入**：

**输出**：（程序会指明出错的行数以及无法识别的符号）



4．（没有正确结束程序,以en结尾）

**输入**：

**输出**：

