编号：

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 实习 | 一 | 二 | 三 | 四 | 五 | 六 | 七 | 八 | 九 | 十 | 总评 | 教师签名 |
| 成绩 |  |  |  |  |  |  |  |  |  |  |  |  |

武汉大学计算机学院

**《编译原理》课程**

语法分析

实习报告

班 序 号： CS42

实习题目： 语法分析

专业（班）： 计算机科学与技术7班

学生学号： 2018302100026

学生姓名： 龙 晓 怡

任课教师： 杜 卓 敏

2021 年 6 月 15 日

目录

[1 语言语法规则 2](#_Toc74130730)

[2 文法定义（消除左递归） 3](#_Toc74130731)

[3 语法分析算法 4](#_Toc74130732)

[3.1 语法分析功能 4](#_Toc74130733)

[3.2自上而下语法分析方法 4](#_Toc74130734)

[3.3递归下降分析方法（递归子程序法） 5](#_Toc74130735)

[3.3.1基本思想 5](#_Toc74130736)

[3.3.2分析过程 5](#_Toc74130737)

[3.3.3 分析子程序构造方法 5](#_Toc74130738)

[4 递归下降语法分析部分关键函数 6](#_Toc74130739)

[递归子程序(状态转换图的实现) 7](#_Toc74130740)

[void s() 7](#_Toc74130741)

[void A() 7](#_Toc74130742)

[Void A\_s() 8](#_Toc74130743)

[Void B() 8](#_Toc74130744)

[Void B\_s() 9](#_Toc74130745)

[Void c() 9](#_Toc74130746)

[Void D() 10](#_Toc74130747)

[Void E() 10](#_Toc74130748)

[Void F() 11](#_Toc74130749)

[Void G() 11](#_Toc74130750)

[Void H() 12](#_Toc74130751)

[Void I() 12](#_Toc74130752)

[Void I\_s() 12](#_Toc74130753)

[Void J() 13](#_Toc74130754)

[Void J\_s() 13](#_Toc74130755)

[Void K() 13](#_Toc74130756)

[Void L() 14](#_Toc74130757)

[Void M() 14](#_Toc74130758)

[5 出错处理出口 15](#_Toc74130759)

[报错函数void RDParser::error(int type) 15](#_Toc74130760)

[6 测试计划 16](#_Toc74130761)

[6.1正确测试用例 16](#_Toc74130762)

[6.2错误测试用例 16](#_Toc74130763)

[6.3 实验结果 17](#_Toc74130764)

[6.3.1词法分析result.dyd输出结果 17](#_Toc74130765)

[6.3.2错误文件error.err输出 18](#_Toc74130766)

# 1 语言语法规则

<程序>→<分程序>

<分程序>→begin <说明语句表>；<执行语句表> end

<说明语句表>→<说明语句>│<说明语句表> ；<说明语句>

<说明语句>→<变量说明>│<函数说明>

<变量说明>→integer <变量>

<变量>→<标识符>

<标识符>→<字母>│<标识符><字母>│ <标识符><数字>

<字母>→a│b│c│d│e│f│g│h│i│j│k│l│m│n│o │p│q │r│s│t│u│v│w│x│y│z

<数字>→0│1│2│3│4│5│6│7│8│9

<函数说明>→integer function <标识符>（<参数>）；<函数体>

<参数>→<变量>

<函数体>→begin <说明语句表>；<执行语句表> end

<执行语句表>→<执行语句>│<执行语句表>；<执行语句>

<执行语句>→<读语句>│<写语句>│<赋值语句>│<条件语句>

<读语句>→read(<变量>)

<写语句>→write(<变量>)

<赋值语句>→<变量>:=<算术表达式>

<算术表达式>→<算术表达式>-<项>│<项>

<项>→<项>\*<因子>│<因子>

<因子>→<变量>│<常数>│<函数调用>

<常数>→<无符号整数>

<无符号整数>→<数字>│<无符号整数><数字>

<条件语句>→if<条件表达式>then<执行语句>else <执行语句>

<条件表达式>→<算术表达式><关系运算符><算术表达式>

<关系运算符> →<│<=│>│>=│=│<>

# 2 文法定义（消除左递归）

改进文法G'如下：

S→begin A B end

A→C;A'

A'→C;A'|ε

C→integer D

D→<标识符> | function <标识符>（<标识符>）；E

E→begin A B end

B→FB'

B'→;FB'|ε

F→read(<标识符>)│write(<标识符>)│G│H

G→<标识符>:=I

I→JI'

I'→-JI'|ε

J→KJ'

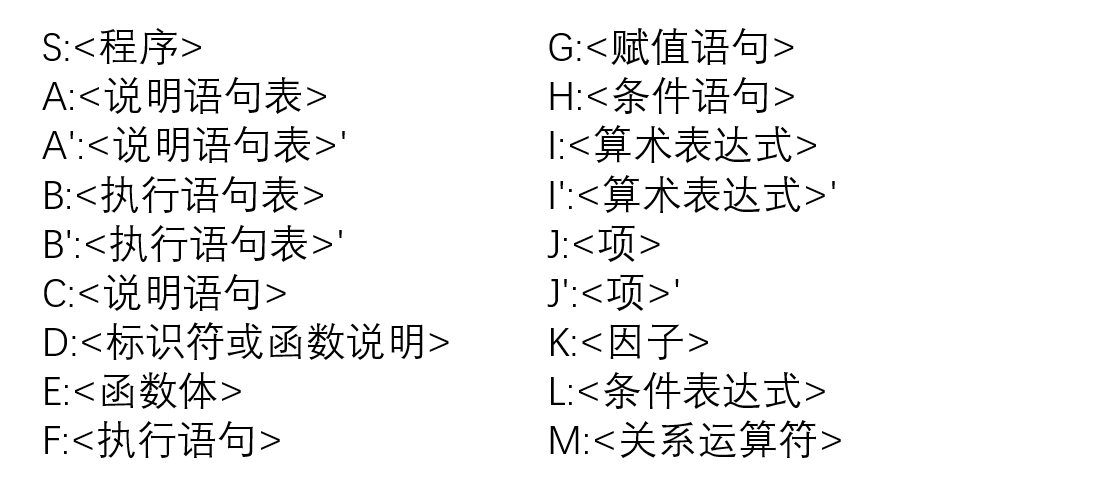
J'→\*KJ' | ε

K→<标识符>│<常数>│<函数调用>

H→if L then F else F

L→IMI

M→<│<=│>│>=│=│<>

其中：

# 3 语法分析算法

## 3.1 语法分析功能

语法的分析是依据语法规则，逐一分析词法分析时得到的单词串，把单词串分解成各类语法单位，即确定它们是怎样组成说明和语句，以及说明和语句又是怎样组成程序的。分析时如发现有不合语法规则的地方，便将出错的位置及出错性质打印报告给程序员；如无语法错误，则用另一种中间形式给出正确的语法结构，供下一阶段分析使用。

## 3.2自上而下语法分析方法

从推导的角度看，从识别符号出发，不断建立直接推导，试图 构造一个最左推导序列，最终推导出与输入符号串相同的符号串。 从语法树的角度看，以识别符号为根结点，试图向下构造一棵 语法树，其末端结点符号串正好与输入符号串相同。 相应于高级语言的编译过程，自上而下语法分析就是从该高 级语言文法的开始符号—— 出发，试图推导得到该文法 的句子—— 源程序或与其等价的单词串

## 3.3递归下降分析方法（递归子程序法）

### 3.3.1基本思想

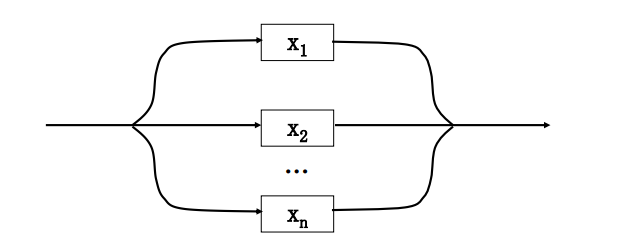
对每一个语法成分（用非终结符号代表），构造相应的分析子程序，该分析子程序分析相应于该语法成分（非终结符号）的符号串。

由于语法成分之间不可避免会含有递归，所以分析子程序之间也会有递归调用，故而又称为递归子程序法

### 3.3.2分析过程

从开始符号出发，在语法规则支配下，逐个扫描输入符号串中的符号，根据文法和当前的输入符号预测到下一个语法成分是U时，便确定U为目标，并调用U的分析子程序P(U)工作。在P(U)工作的过程中，又有可能确定U或其它非终结符号为子目标，并调用相应的分析子程序。如此继续下去，直到得到结果。

### 3.3.3 分析子程序构造方法

1）对于每个非终结符号U，编写一个相应的子程序P(U)。

2）对于产生式U→x1 |x2 |…|xn，有一个关于U的子程序P(U) 。U可空

IF CH IN FIRST(x1 ) THEN P(x1 )

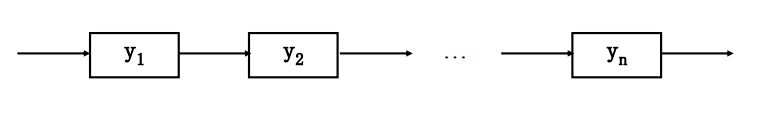
ELSE IF CH IN FIRST(x2 ) THEN P(x2 )

ELSE …

…

IF CH IN FIRST(xn ) THEN P(xn )

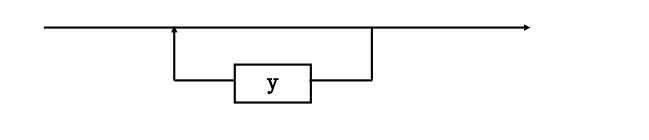
ELSE IF not(CH IN FOLLOW(U)) THEN ERROR

3）对于x=y1y2…yn； BEGIN P(y1 )；P(y2 )；…；P(yn ) END 。

如果①yi∈VN，则P(yi)就代表调用处理yi的子程序；

②yi∈VT，则P(yi)为形如下述语句的一段程序

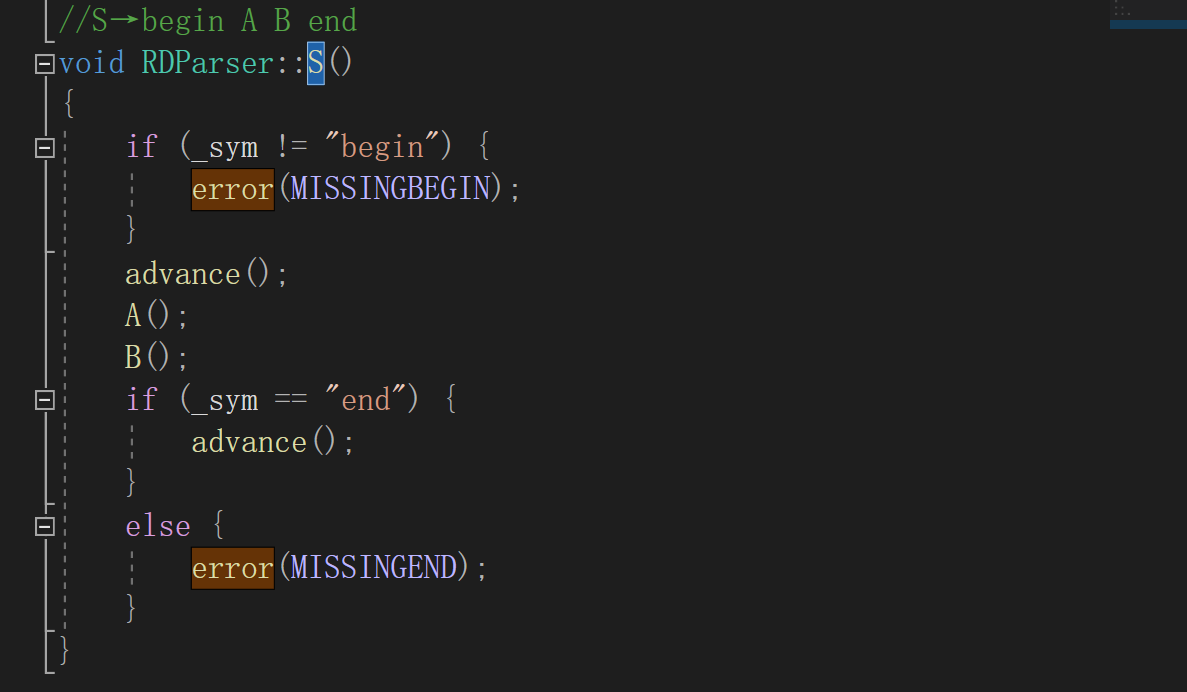
IF CH=yi THEN READ(CH) ELSE ERROR

4）如果x={y}，在程序中就是一个循环。

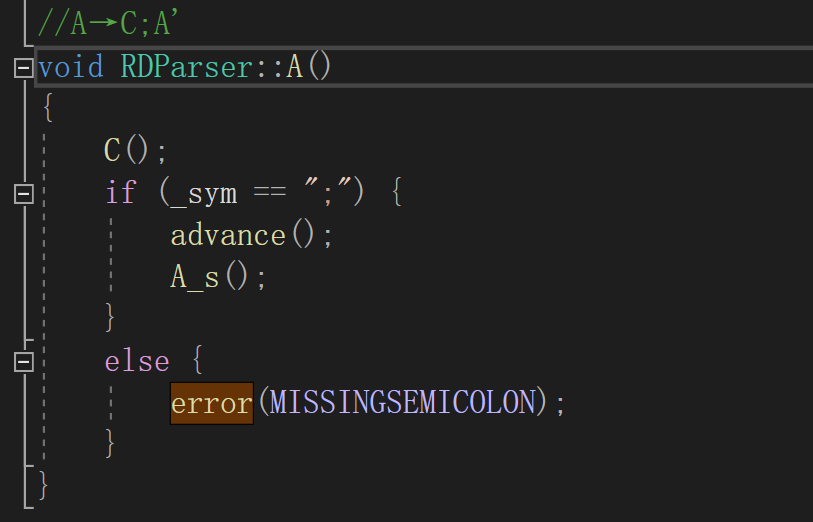
# 4 递归下降语法分析部分关键函数

## 递归子程序(状态转换图的实现)

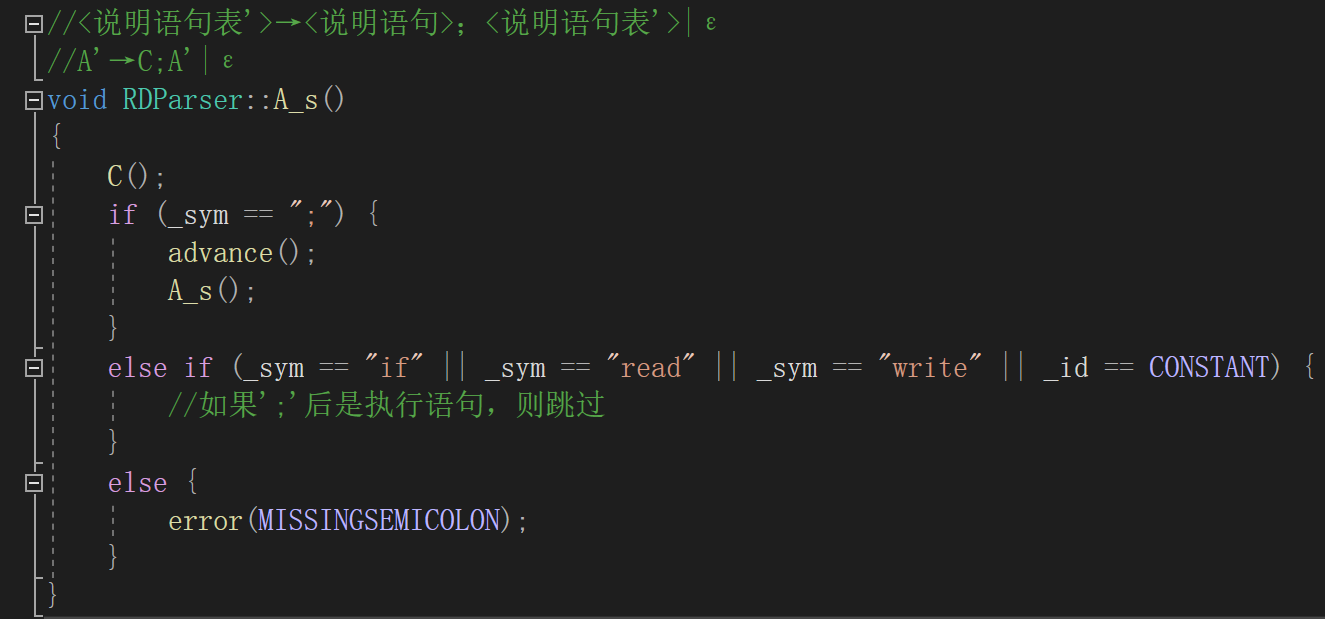
### void s()

<程序>→begin <说明语句表><执行语句表> end

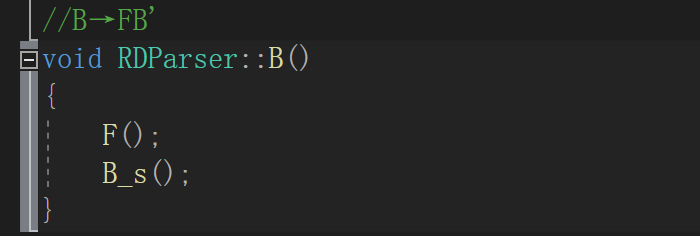
### void A()

<说明语句表>→<说明语句>；<说明语句表'>

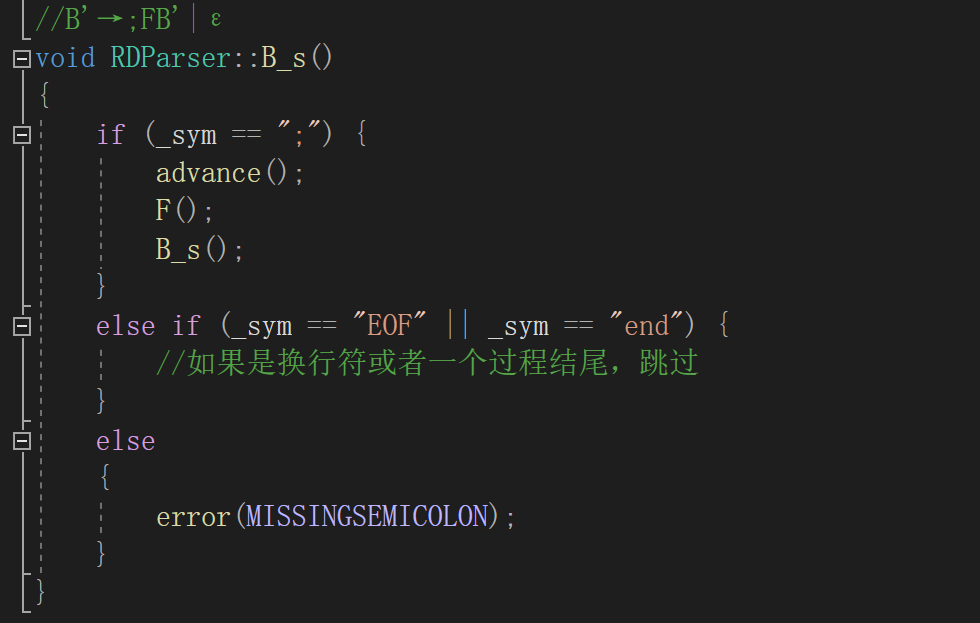
### Void A\_s()

<说明语句表'>→<说明语句>；<说明语句表'>|ε

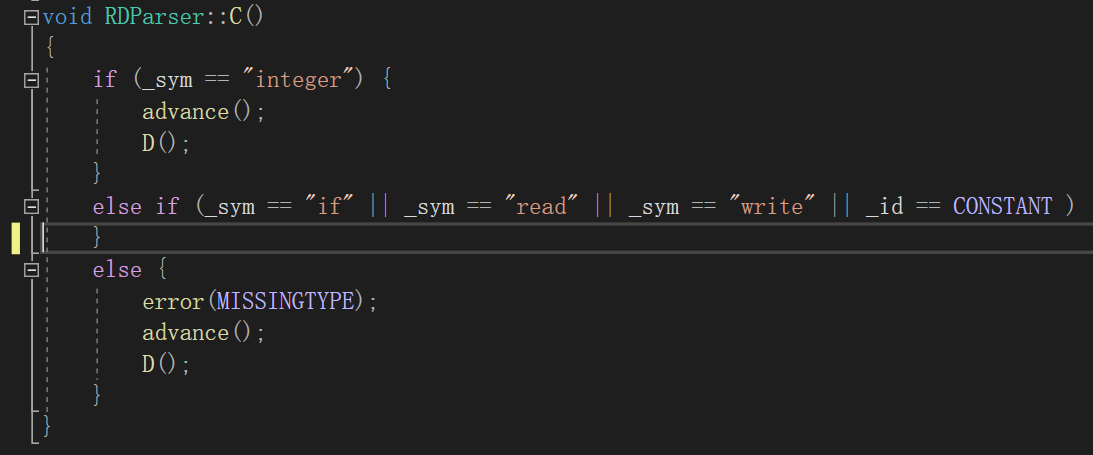
### Void B()

<执行语句表>→<执行语句><执行语句表'>

### Void B\_s()

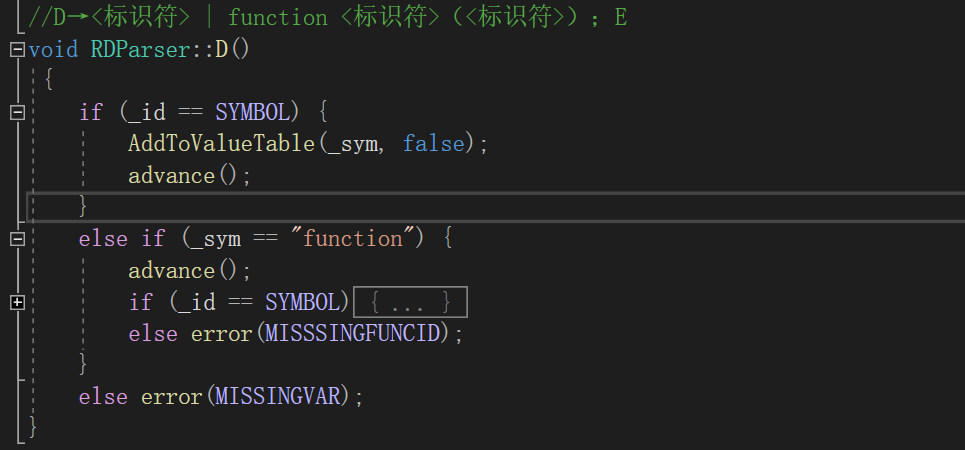
<执行语句表'>→; <执行语句><执行语句表'> | ε

### Void c()

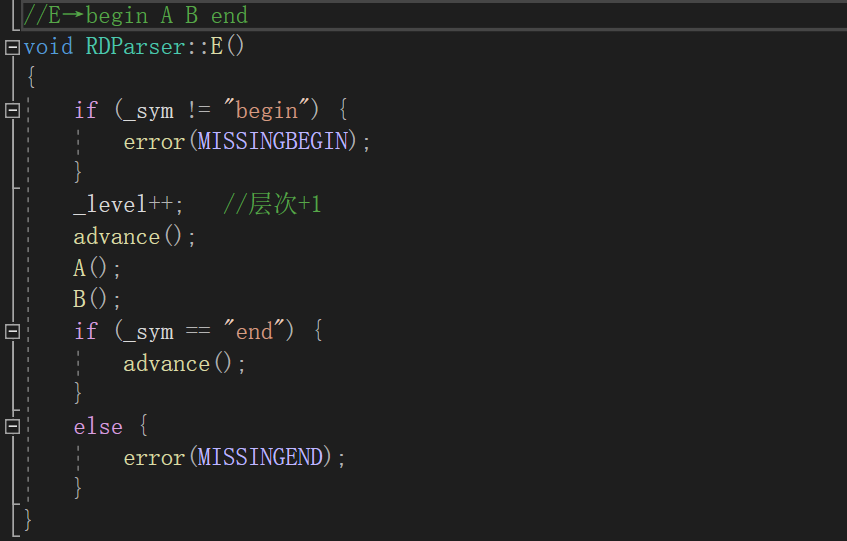
<说明语句>→integer <标识符或函数说明>

### Void D()

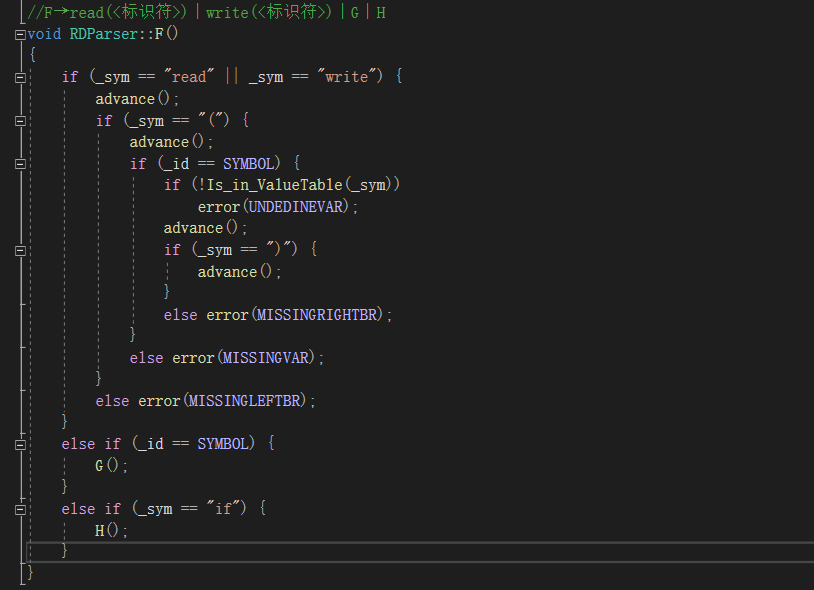
D→<标识符> | function <标识符>（<标识符>）；E

AddToValueTable（）：添加变量到变量名表中

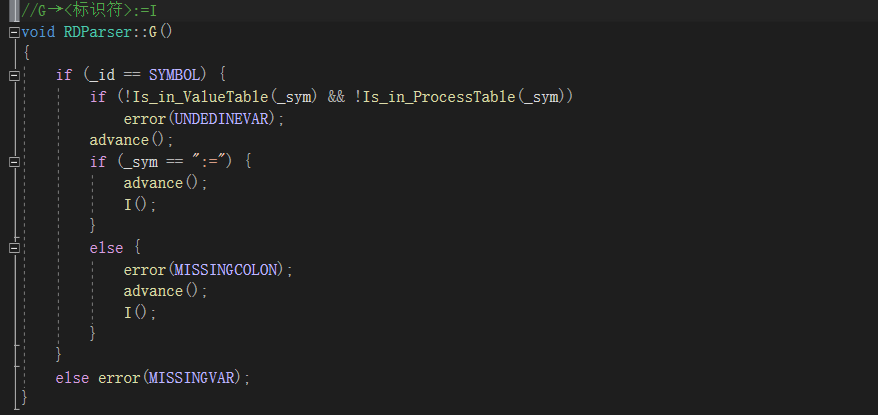
### Void E()

<函数体>→begin <说明语句表><执行语句表> end

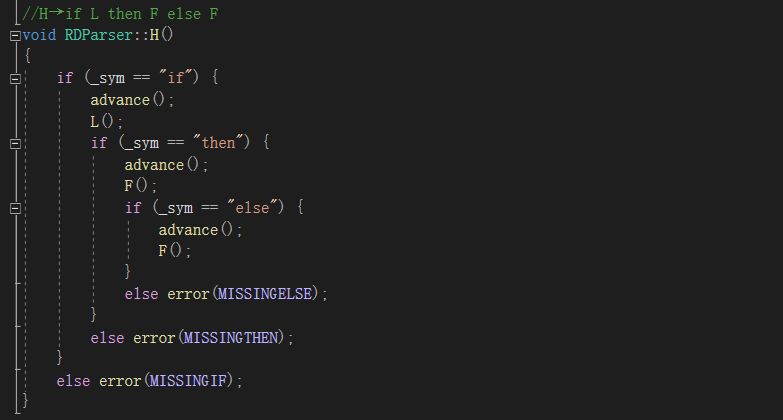
### Void F()

<执行语句>→read(<标识符>)│write(<标识符>)│<赋值语句>│<条件语句>

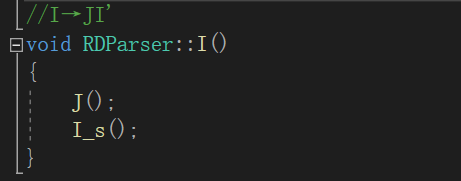
### Void G()

<赋值语句>→<标识符>:=<算术表达式>

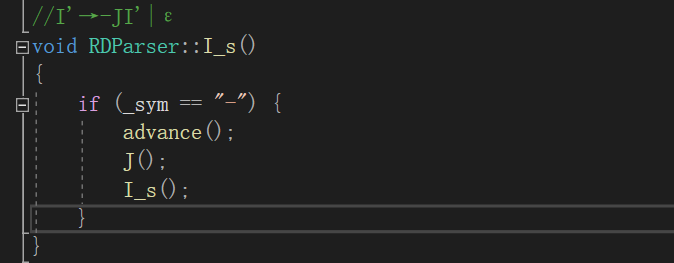
### Void H()

<条件语句>→if<条件表达式>then<执行语句>else <执行语句>

### Void I()

<算术表达式>→<项><算术表达式>'

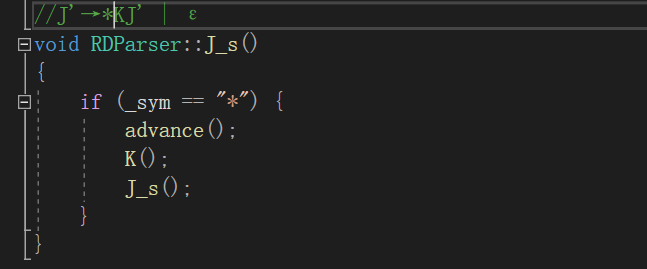
### Void I\_s()

<算术表达式>'→-<项><算术表达式>'

### Void J()

<项>→<因子><项'>

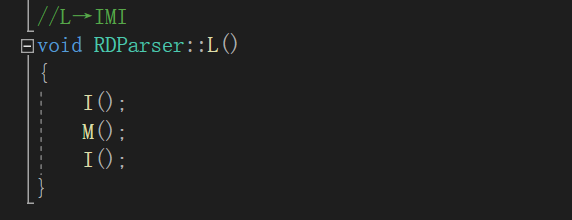
### Void J\_s()

<项'>→\*<因子><项'> | ε

### Void K()

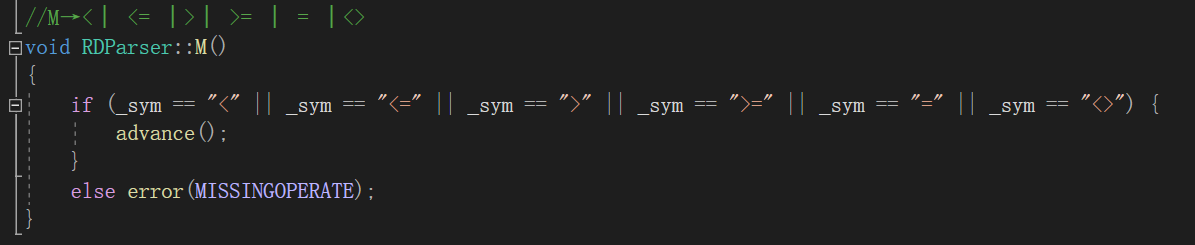
<因子>→<标识符>│<常数>│<函数调用>

### Void L()

<条件表达式>→<算术表达式><关系运算符><算术表达式>

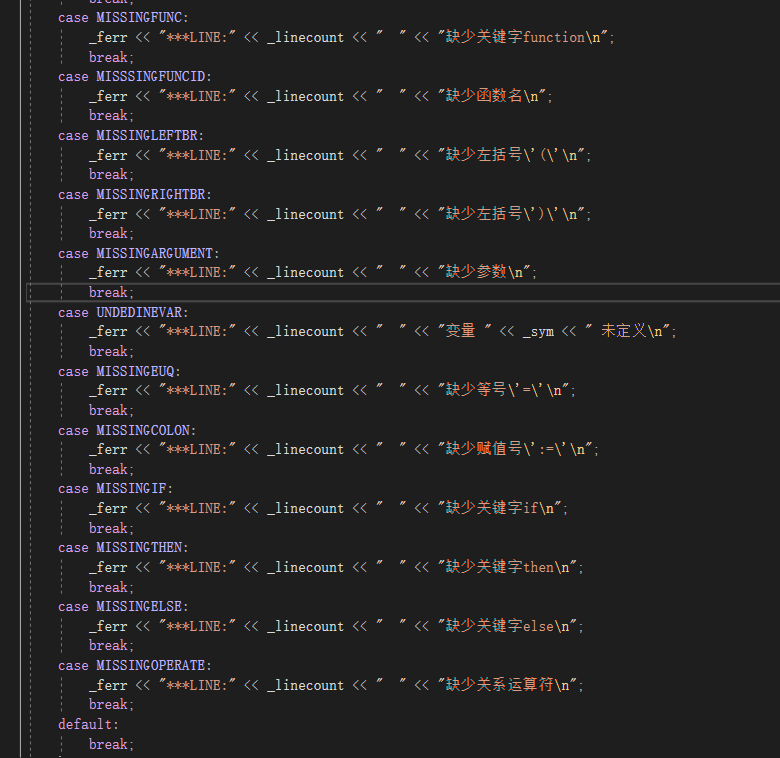
### Void M()

<关系运算符> →<│<=│>│>=│=│<>



# 5 出错处理出口

## 报错函数void RDParser::error(int type)

语法分析器根据传入类型报错

# 6 测试计划

求n！（该测试程序包含算术表达式，逻辑表达式，赋值语句，if-then分支结构，while循环结构）

## 6.1正确测试用例

## 6.2错误测试用例

本测试用例中设计了以下错误：

Begin end不匹配

Interger 标识符错误

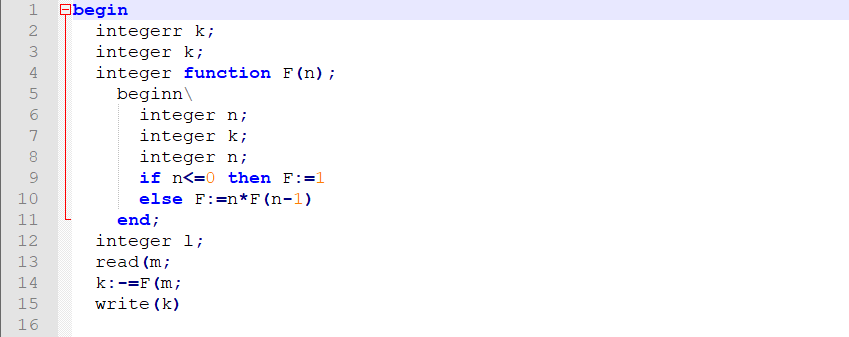
Begin 标识符错误

非法字符\

重复定义变量k, n

未定义变量m

缺少括号）、缺少赋值符号：=

****

## 6.3 实验结果

### 6.3.1词法分析result.dyd输出结果

每行结束输出EOLN 24, 文件结束输出EOF 25

### 6.3.2语法分析错误文件error.err输出