# 文法解读

1. <程序>      ::=  <分程序>.  
   <分程序>  ::=   [<常量说明部分>][<变量说明部分>]{[<过程说明部分>]| [<函数说明部分>]}<复合语句>

·分析：程序以分程序内容后“.”为标志结束。分程序分为常量说明、变量说明、过程说明或函数说明、复合语句四大部分。其中常量说明和变量说明分别最多出现一次，不能重复说明，函数说明或者过程说明可以出现多次，复合语句是不可以省略的部分。分程序的这几个部分有着出现的先后顺序，不可以颠倒。

1. <常量说明部分>  ::=  const<常量定义>{,<常量定义>};  
   <常量定义>  ::=  <标识符>＝ <常量>

<标识符>  ::=  <字母>{<字母>|<数字>}  
<常量>   ::=  [+| -] (<无符号整数>| <无符号实数>)|<字符>

<无符号整数>   ::=  <数字>{<数字>}

<无符号实数>  ::=  <无符号整数>.<无符号整数>

<字符>   ::=  '<字母>' | '<数字>'

·分析：常量说明部分以“const”作为开始标志，以“；”作为结束标志。一个常量说明部分至少出现一个常亮定义，如果出现多个，用逗号隔开。注意常量定义用等号来链接标识符和值。标识符必须以字母开头，字母后可以链接多个字母或者数字。常量可以有三种，无符号整数、无符号实数和字符。其中无符号整数和无符号实数之前可以有“+”或“-”，字符需要单引号引住。根据文法，无符号整数可以零起。

·举例：const c1=+000, c2=-001406, c3=+123, c4=-123, c5=’a’, c6=’1’, c7=’0’;

1. <变量说明部分>::=  var <变量说明> ; {<变量说明>;}  
   <变量说明>  ::=  <标识符>{, <标识符>} : <类型>  
   <类型>   ::=   <基本类型>|array'['<无符号整数>']' of <基本类型>  
   <基本类型>  ::=   integer | char | real

·分析：变量说明部分以“var”作为开始标志，“变量说明”至少出现一次。一个变量说明可以说明多个同一类型的变量。类型可以为基本的三个类型：整数、实数和字符，也可以为基本类型的数组，数组必须定义的长度，且长度可以为零。

·举例：var i1: integer;

c1, c2: char;

r1, r2, r3: real;

ia1: array[0] of integer;

ca1: array[10] of char;

1. <过程说明部分>  ::=   <过程首部><分程序>{;<过程首部><分程序>};  
   <函数说明部分>  ::=   <函数首部><分程序>{;<函数首部><分程序>};  
   <过程首部> ::=  procedure<标识符>[<形式参数表>];  
   <函数首部>  ::=  function <标识符>[<形式参数表>]: <基本类型>;  
   <形式参数表>   ::= '(' <形式参数段>{; <形式参数段>}')'  
   <形式参数段>::=  [var]<标识符>{, <标识符>}: <基本类型>

·分析：过程说明部分包括至少一个过程首部和分程序，以分号结束，这里出现一个递归。过程首部以procedure开始，函数首部以function开始，后面跟着标识符，可能跟着一个形式参数表，函数首部还比过程首部多一个基本类型，代表函数返回的类型。形式参数表被括号括起来其中包括若干个形式参数段，以分号间隔。形式参数段开头可有一个var标志，后面至少一个或多个标识符，逗号隔开，再后面是基本类型，表示参数的数据类型。

·举例：procedure p1 (var i1, i2: integer; c1, c2: char)

const ia1: array[0] of integer;

var ca1: array[5] of char;

function f1 (var r1, r2: real; var i3,i4:integer): integer

begin

(程序段)

end;

begin

(程序段)

end;

1. <语句> ::=  <赋值语句>|<条件语句>|<情况语句>|<过程调用语句>|<复合语句>|<读语句>|<写语句>|<for循环语句>|<空>  
   <赋值语句> ::=  <标识符> := <表达式>| <函数标识符> := <表达式> | <标识符>'['<表达式>']':= <表达式>

<条件语句> ::=  if<条件>then<语句> | if<条件>then<语句>else<语句>

<情况语句> ::=  case <表达式> of <情况表元素>{; <情况表元素>}end

<情况表元素>   ::=  <常量> : <语句>

<过程调用语句> ::=  <标识符>[<实在参数表>]

<复合语句>   ::=  begin<语句>{; <语句>}end  
<读语句> ::=   read'('<标识符>{,<标识符>}')'  
<写语句> ::=   write '(' <字符串>,<表达式> ')'|write'(' <字符串> ')'|write'('<表达式>')'

<for循环语句> ::=  for <标识符>  := <表达式> （downto | to） <表达式> do <语句>  //步长为1

·分析：语句分为9种，赋值、条件语句，情况、过程调用、复合语句、度语句、写语句，for循环，和空语句。由于很多这几种语句中都有<语句>成分，条件语句和复合语句中存在着递归的情况。

·举例：赋值：a:=b\*c+d-e

F1:=f-g\*c+d

A[1+2-3\*4]=233

条件：if a>b then

if c>d then

a:=c

else

a:=d

情况：case a of

1:

2:

3:

End

过程调用：P1(a,b,c,d)

复合语句：begin begin begin ;;;;;;; end end end

读语句：read(a,a,a,a,a,a,a,a,)

写语句：write(“abcdef”)

Write(a)

Write(“write”,a+b)

For循环：for a=0 to 10 do

For b=0 to 10 do begin

;;;;

End

1. <表达式> ::= [+|-]<项>{<加法运算符><项>}  
   <项>::= <因子>{<乘法运算符><因子>}  
   <因子>::= <标识符>|<无符号整数>| <无符号实数>|'('<表达式>')' | <函数调用语句>|<标识符>'['<表达式>']'  
   <函数调用语句>::=<标识符>[<实在参数表>]  
   <实在参数表>::= '('<实在参数> {, <实在参数>}')'  
   <实在参数>::=<表达式>  
   <加法运算符>::= +|-  
   <乘法运算符>::=\*|/  
   <条件>::=<表达式><关系运算符><表达式>  
   <关系运算符> ::= <|<=|>|>= |=|<>

<函数标识符>::=<标识符>  
<字母> ::=   a|b|c|d|…|x|y|z |A|B|…|Z  
<数字>    ::=   0|1|2|3|…|8|9

·分析：表达式可以由加法运算符开头，由多个项和加法运算符链接，项由因子开头，包含至少一个因子，多个因子由乘法运算符链接。因子由标识符，无符号整数，无符号实数，带括号的表达式，函数调用语句，数组的元素组成。其中函数调用语句指的是函数的返回值。条件由表达式和关系运算符组成，条件可以返回一个整型或者布尔类型的值。字母包含26个英文字母的大小写。数字0~9。

·举例：表达式：+a\*(-b) +000123+-0001112.3334- F( a,b,-234)+A[F(1+2+0.5,0,0)]-000