# 文法解读

撰写一份解读文档，要求对文法中每条规则所定义的语法成分进行分析，说明其作用、限定条件，给出句子示例。

**1、**<程序> :: =   <分程序>.

**范例：**

const a = 6;.

**分析：**程序由分程序组成，程序结尾以“.”进行标识。

**2、**<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>

<常量说明部分>  ::=   const<常量定义>{,<常量定义>};

<常量定义>           ::=    <标识符>＝ <常量>

<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>

<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>;

**范例：**

const a = 1406;//常量说明

var b:integer;//变量说明

procedure id;//过程说明

id(新的分程序);

function getid:interger;//函数说明，可带参数

getid(新的分程序）;

begin//复合语句开始

分程序的语句部分，略

End//复合语句结束，begin和end都没有分号

**分析：**

分程序由常量说明、变量说明、过程说明、函数说明和复合语句几个部分组成。除复合语句外其他部分均可以为空，但是在一个分程序中不会重复出现，即应一次性完成一个分程序的常量说明等过程。且各个部分应当按顺序给出，不能出现诸如先进行变量声明再进行常量说明的情况。

变量说明和函数说明中应给出变量的数据类型以及函数返回值的数据类型。过程和函数都是由相应的子程序组成的。函数的主题是由一对begin和end标记出来的复合语句。

**3、**<常量>             ::=    [+| -] (<无符号整数>| <无符号实数>)|<字符>

<字符>                   ::=  '<字母>' | '<数字>'

<字符串>               ::= "{十进制编码为32,33,35-126的ASCII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>

<标识符>               ::=   <字母>{<字母>|<数字>}

<类型>                 ::=   <基本类型>|array'['<无符号整数>']' of <基本类型>

<基本类型>      ::=   integer | real| char

<字母>                   ::=   a|b|c|d&hellip;x|y|z |A|B&hellip;|Z

<数字>                   ::=   0|1|2|3&hellip;8|9

**范例：**

05 +06 -7 //无符号整数

5.0 +6.02 -7.880//无符号实数

十进制编码为32,33,35-126的ASCII字符

如’a’ # //字符

“Buaa” “@edu” //字符串

a aaaa student0//标识符

integer array[5]of char//类型

**分析：**

标识符是以字母（ascii十进制编码在65-90和97-122的字符）开头，后接字母或数字（ascii十进制编码48-57的字符）的一个字符串，文法中没有限制标识符的长度。重点是字符没有引号。

常量包括数字和字符，其中数字包括整数和实数，整数和实数又分为有符号整数（实数）和无符号整数（实数）。**这里没有找到有关bnf运算符优先级的内容，认为字符前不会出现“+”或“-”。**

无符号整数是由数字组成的字符串，在无符号整数前加上“+”或“-”就得到了整数，无符号整数也是整数；无符号实数是由两个由数字组成并以小数点“.”分隔的字符串组成的，在无符号实数前加上“+”或“-”就得到了实数，无符号实数也是实数。数字前导零均为合法，数字结尾的零也合法。

共有整型（integer）、实型（real）和字符型（char）三种基本的数据类型，并对应有三种类型的数组，声明数组时应给出数组的类型和数组的大小，仅支持一维数组。

**4、**<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>

<实在参数表>      ::= '(' <实在参数> {, <实在参数>}')'

<实在参数>           ::=   <表达式>

**范例：**

**略。**

**分析：**

函数说明和过程说明中包括形式参数表，函数和过程调用时包含实在参数表，各个参数以都好隔开，参数表用小括号标出。

形式参数表需要给出形式参数的数据类型，数组不能当形式参数，若形式参数为变量需要标明。实在参数为表达式，以逗号隔开。

5、<函数标识符>   ::=  <标识符> （好像应该把它和标识符部分放在一起）

<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>|<标识符>'['<表达式>']'|<无符号整数>| <无符号实数>| '('<表达式>')' | <函数调用语句>

<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>      ::=   <|<=|>|>= |=|<>

**范例：**

A (a+b) 123.456 a[0]//因子

5\*a 30 6\*5 //项

A+b //表达式

**分析：**

加法运算符是加法运算符“+”和减法运算符“-”，乘法运算符是乘法运算符“\*”和除法运算符“/”。

表达式是项的和，项是因子的积，单个的因子也是项和表达式，表达式起始可以有加法运算符。

条件由表达式之间的关系表示。表达式中不应该出现函数标识符， 函数标识符在文法中仅出现了一次，**这里不清楚是否是所有标识符都可以替换为函数标识符，但是考虑到函数是有自己的数据类型的所以我觉得是可以替换的。**

6、 <语句>            ::=   <赋值语句>|<条件语句>|<当循环语句>|<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>

<函数调用语句>     ::=   <标识符>[<实在参数表>]

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

<当循环语句> ::= do<语句> while<条件>

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

<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>')'

**范例：**

read(a,b); //读语句

write(“buaa”,a+b); //写语句

id = 1186; //赋值语句

if(1>0)

then

else //条件语句

do

while //do-while循环语句

for 1=10 to 100

do //for循环语句

**分析：**语句可以是空语句，复合语句是由begin和end标注的至少一个语句，故复合语句begin和end中间的部分可以为空。

语句一共有赋值语句、函数调用语句、条件语句、循环语句、过程调用语句、读语句和写语句七类。各类语句都不是以分号结尾的，最后的作业中需要对语句的划分部分进行较为完备的设计。

读语句应给出想要读取的变量名

写语句可以一次性输出一个字符串或一个表达式的值或字符串和表达式的值，不能是多个。

赋值语句可以对普通变量、数组项和函数进行赋值，常量说明部分也可以认为是赋值语句，但是不利于进行语义分析。

条件句的else分支可以省略，then分支和else分支都可以为空

循环语句有两类，一类是do-while型，没啥特别的，另一类是for型，主义步长的变化方向由downto和to决定，to是增加downto是减少。