<程序>      ::=  <分程序>.  
 {\*程序由分程序组成\*}  
<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>  
 {\*分程序由可选的常量说明、变量说明、过程说明及函数说明开始，随后是复合语句\*}  
<常量说明部分>   ::=  const<常量定义>{,<常量定义>};  
 {\*常量说明前面要加const，后面可跟多个用逗号分隔的常量定义，以分号结尾\*}  
 【\* const ARRLEN = 10, ENDCHAR = '@' \*】  
<常量定义>    ::=  <标识符>＝ <常量>  
 {\*常量定义的格式：用等号进行定义\*}  
<常量>      ::=  [+| -]<无符号整数>|<字符>  
 {\*常量为整数或字符\*}  
<字符>      ::=   '<字母>' | '<数字>'  
 {\*字符是用单引号括起来的单个字母或单个数字\*}  
<字符串>     ::=   "{十进制编码为32,33,35-126的ASCII字符}"  
 {\*字符串是双引号括起来的，可为一个或多个的可打印字符\*}  
<无符号整数>   ::=  <数字>{<数字>}  
<标识符>     ::=  <字母>{<字母>|<数字>}  
 {\*标识符是又字母打头的，字母和数字的任意组合\*}  
<变量说明部分>  ::=  var <变量说明> ; {<变量说明>;}  
 {\*用var作前缀表示一组变量说明，每个变量说明之后用分号作为结束标识\*}  
<变量说明>    ::=  <标识符>{, <标识符>} : <类型>    
 {\*一个变量说明可以声明同一类型的多个变量，类型前用冒号进行标识，不同变量的标识符之间用逗号隔开\*}  
<类型>      ::=  <基本类型> | array '['<无符号整数>']' of  <基本类型>  
 {\*类型可以是基本类型或数组类型\*}  
<基本类型>    ::=  integer | char   
 {\*基本类型为整数或字符\*}  
<过程说明部分>  ::=  <过程首部><分程序>{;<过程首部><分程序>};  
 {\*过程说明分为若干个过程首部和分程序\*}  
<函数说明部分>  ::=  <函数首部><分程序>{;<函数首部><分程序>};  
 {\*函数说明分为若干个函数首部和分程序\*}  
<过程首部>    ::=   procedure<标识符>'('[<形式参数表>]')';  
 {\*过程首部为procedure加标识符加括号括起来的，中间内容可选的形式参数表\*}  
<函数首部>    ::=   function <标识符>'('[<形式参数表>]')': <基本类型>;  
 {\*函数首部为function加标识符加括号括起来的，中间内容可选的形式参数表，再加冒号以及基本类型，表示函数的返回值类型，以分号结束\*}  
<形式参数表>   ::=  <形式参数段>{; <形式参数段>}  
 {\*形式参数表由一个或多个形式参数段组成\*}  
<形式参数段>   ::=  [var]<标识符>{, <标识符>}: <基本类型>  
 {\*形式参数段由可选的var标志，以及用逗号分隔的若干个标识符，以及用冒号分隔的一个基本类型组成\*}  
<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>  
 {\*语句由赋值语句条件语句或情况语句或过程调用语句或复合语句或读语句或写语句或for循环语句组成，也可为空\*}  
<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>  
 {\*赋值语句的核心是:=这个赋值符号，赋值符号右边是表达式，赋值符号左边可以是标识符、函数标识符或数组元素\*}  
<函数标识符>   ::=  <标识符>   
 {\*函数标识符的词法结构和标识符是一样的\*}  
<表达式>     ::=  [+|-]<项>{<加法运算符><项>}  
 {\*表达式由多项相加减而成，首项可以有正负号\*}  
<项>       ::=  <因子>{<乘法运算符><因子>}  
 {\*每一项由一个或多个因子相乘/除组成\*}  
<因子>      ::=   <标识符>|<无符号整数> |'('<表达式>')' | <函数调用语句>|<标识符>'['<表达式>']'  
 {\*每个因子由标识符或无符号整数或括号括起来的表达式或函数调用语句或数组元素构成\*}  
<函数调用语句>  ::=   <标识符>'('[<实在参数表>]')'  
 {\*函数调用语句由标识符加括号加可选的实在参数表构成\*}  
<实在参数表>   ::=  <实在参数> {, <实在参数>}  
 {\*实在参数表由一个或多个用逗号分隔的实在参数组成\*}  
<实在参数>    ::=   <表达式>  
 {\*实在参数由一个表达式表示\*}  
<加法运算符>   ::=  +|-  
 {\*加法运算符为加号减号\*}  
<乘法运算符>   ::=  \*|/  
 {\*乘法运算符为乘号或除号\*}  
<条件>      ::=  <表达式><关系运算符><表达式>  
 {\*条件为两个用关系运算符连接起来的表达式构成\*}  
<关系运算符>   ::=  <|<=|>|>= |=|<>  
 {\*关系运算符为小于、小于等于、大于、大于等于、等于或不等号\*}  
<条件语句>    ::=  if<条件>then<语句> | if<条件>then<语句>else<语句>  
 {\*条件语句为if条件then语句构成，后面可以接一条else语句\*}  
<情况语句>    ::=  case <表达式> of <情况表元素>{; <情况表元素>}end  
 {\*情况语句由case加表达式加一个或多个of情况表元素和最后的一条end组成\*}  
<情况表元素>   ::=  <情况常量表> : <语句>  
 {\*情况表元素由情况表常量加冒号加语句组成\*}  
<情况常量表>   ::=  <常量>{, <常量>}  
 {\*情况常量表由若干个由逗号分隔的常量组成\*}  
<for循环语句>   ::=  for <标识符>  := <表达式> （downto | to） <表达式> do <语句>    
 {\*for循环语句的格式...\*}  
<过程调用语句>   ::=   <标识符>'('[<实在参数表>]')'  
 {\*过程调用语句由标识符加括号加括号中可选的实在参数表组成\*}  
<复合语句>    ::=  begin<语句>{; <语句>}end  
 {\*复合语句由一对begin和end以及中间一条或多条以分号分隔的语句构成\*}  
<读语句>     ::=  read'('<标识符>{,<标识符>}')'  
 {\*读语句由read加括号加括号中一个或多个标识符构成\*}  
<写语句>     ::=   write'(' <字符串>,<表达式> ')'|write'(' <字符串>')'|write'('<表达式>')'  
 {\*写语句由write加括号加括号中的字符串，括号中的表达式或括号中的字符串、逗号和表达式构成\*}  
<字母>      ::=   a|b|c|d…x|y|z |A|B…|Z  
 {\*字母由26个小写字母以及对应的大写字母构成\*}  
<数字>      ::=   0|1|2|3…8|9  
 {\*数字由0~9这10个数字组成\*}

附加说明：

（1）char类型的变量或常量，用字符的ASCII码对应的整数参加运算

（2）标识符区分大小写字母

（3）赋值语句中<函数标识符> := <表达式> 作为函数的返回值，其类型应与返回类型一致，此语句后面的语句可继续执行

（4）写语句中的字符串原样输出，表达式只有单个字符类型的变量或常量按字符输出，其他表达式均按整型输出

（5）情况语句中，case后面的表达式和情况常量表里面的常量只允许出现integer和char类型

（6）数组的下标从0开始

（7）for循环语句步长为1

（8）带var的形式参数为变量形参，实参与该类形参传递数据时是传地址

{\*需要注意的是，在一组复合语句中，不同语句间要用分号隔开。最后一条语句的后面，可以加分号，也可以不加分号。\*}