<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>  
<常量说明部分>   ::=  const<常量定义>{,<常量定义>};  
<常量定义>    ::=  <标识符>＝ <常量>@ADD2TOKENTABLE  
<常量>      ::=  [+| -]<无符号整数>|<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}"@ADD2STRINGTABLE  
<无符号整数>   ::=  <数字>{<数字>}  
~~<标识符>     ::=  <字母>{<字母>|<数字>}~~  
<标识符>     ::=  <字母>{<字母>|<数字>|’\_’}  
~~<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>~~

<变量说明部分>  ::=  var <变量定义> ; {<变量定义>;}  
<变量定义>    ::=  <标识符>{, <标识符>} : <类型>@ADD2TOKENTABLE  
~~<类型>      ::=  <基本类型> | array '['<无符号整数>']' of  <基本类型>~~  
<类型>      ::=  [array '['<无符号整数>']' of ] <基本类型>  
<基本类型>    ::=  integer | char   
~~<过程说明部分>  ::=  <过程首部><分程序>{;<过程首部><分程序>};~~  
<过程说明部分>  ::=  @BEGINPROCEDURE\_NAME<过程首部><分程序>;@END{@BEGINPROCEDURE\_NAME<过程首部><分程序>;@END}  
~~<函数说明部分>  ::=  <函数首部><分程序>{;<函数首部><分程序>};~~  
<函数说明部分>  ::=  @BEGINFUNCTION\_NAME <函数首部><分程序>;@END{@BEGINFUNCTION\_NAME <函数首部><分程序>;@END }  
<过程首部>    ::=   procedure<过程标识符>'('[<形式参数表>]')';  
<函数首部>    ::=   function <函数标识符>'('[<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>@ADD2TOKENTABLE  
~~<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>      ::=  <标识符>(<赋值语句>|<过程调用语句>|<函数调用语句>)|<条件语句>|<情况语句>|<复合语句>|<读语句>|<写语句>|<for循环语句>|<while循环语句>| <循环继续语句>|<循环退出语句>|<空>  
<循环继续语句> ::= continue @JMPLabel<vary/check>  
<循环退出语句> ::= break @JMPLabel<end>  
~~<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>    ::=  ['['<表达式>']'] := <表达式>@ASG ~~<函数标识符>   ::=  <标识符>   
<过程标识符>   ::=  <标识符>~~  
<表达式>     ::=  [+|-]<项>@NEG{<加法运算符><项>@ADD}  
<项>       ::=  <因子>{<乘法运算符><因子>@MUL}  
~~<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>']'~~<因子>      ::=   <标识符>(['['<表达式>']']|[<函数调用语句>@CALL\_FUNC]) | '('<表达式>')' | <无符号整数>|字符  
~~<函数调用语句>  ::=   <标识符>'('[<实在参数表>]')'~~

<函数调用语句>  ::=   '('[<实在参数表>]')'@FUNC\_CALL  
~~<实在参数表>   ::=  <实在参数> {, <实在参数>}  
<实在参数>    ::=   <表达式>~~<实在参数表>   ::=  <表达式>@SETP {, <表达式>@SETP}  
<加法运算符>   ::=  +|-  
<乘法运算符>   ::=  \*|/  
~~<条件>      ::=  <表达式><关系运算符><表达式>~~  
~~<条件>      ::=  <表达式>[<关系运算符><表达式>]~~  
<条件> ::= <布尔表达式>  
<布尔表达式> ::= <布尔项>@Label<endterm> [<逻辑或> <布尔项>@Label<endterm>] @JMP<label\_negative>  
<布尔项> ::= <布尔因子>@Label<endfactor> [<逻辑与><布尔因子>@Label<endfactor>] @JMP<label\_positive>  
<布尔因子> ::= [<逻辑非>]<表达式>@JE<Label\_negative>  
<布尔因子> ::= [<逻辑非>]<表达式><关系运算符><表达式>@JZ<Label\_negative>  
<布尔因子> ::= '('<布尔表达式>')'  
<关系运算符>   ::=  <|<=|>|>= |=|<>  
<逻辑或> ::= ||  
<逻辑与> ::= &&  
<逻辑非> ::= !  
 <条件语句> ::= if<条件>@LABEL\_beginthen then<语句>@LABEL\_endthen  
 <条件语句> ::= if<条件>@LABEL\_beginthen then<语句>@JMP\_endelse @LABEL\_endthen else<语句>@LABEL\_endelse  
<情况语句>    ::=  case <表达式>@JELABEL<i> @JMP<END>of <情况表元素>@JMP<END>{; <情况表元素>}end @LABEL<END>  
<情况表元素>   ::=  <情况常量表>@LABEL<i> : <语句>  
~~<情况常量表>   ::=  <常量>{, <常量>}~~<情况常量表>   ::=  <常量|标识符>{, <常量|标识符>} 【语义要求：这里的标识符需要为常量定义处所定义的标识符】  
<for循环语句>   ::=  for <标识符>  := <表达式>（downto | to）@ASG<init> @JMPLABEL<check> @Label<vary>@ASG<vary> @Label<check> <表达式> @JZLABEL<end> do <语句>@JMPLABEL<vary>@Label<end> 【语义要求：这里的标识符不能为函数名<区别于函数的返回语句>】  
<while循环语句> ::= while @Label<check> <条件> do @Label<BeginDo> <语句> @JMPLabel<check> @Label<end>  
~~<过程调用语句>   ::=   <标识符>'('[<实在参数表>]')'~~<过程调用语句>   ::=   '('[<实在参数表>]')'@PROC\_CALL  
<复合语句>    ::=  begin<语句>{; <语句>}end  
~~<读语句>     ::=  read'('<标识符>{,<标识符>}')'~~  
<读语句>     ::=  read'('<标识符>[‘[’<表达式>’]’] @READ {,<标识符>[‘[’<表达式>’]’] @READ }')'  
~~<写语句>     ::=   write'(' <字符串>,<表达式> ')'|write'(' <字符串>')'|write'('<表达式>')'~~  
<写语句>     ::=   write'(' (<字符串>@WRITE [,<表达式>@ WRITE] | <表达式>@ WRITE) ')'  
<字母>      ::=   a|b|c|d…x|y|z |A|B…|Z  
<数字>      ::=   0|1|2|3…8|9

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