<程序> → <变量说明>BEGIN<语句表>END.

<变量说明> → VAR<变量说明序列>

<变量说明序列> → <变量表>:INTEGER; <变量说明序列公因式>|ε

<变量说明序列公因式> → ε|<变量说明序列>

<变量表> → ID|ID,<变量表>

<语句表> → <语句><语句公因式>|ε

<语句公因式> → ε | <语句表>

<语句> → <赋值语句>|<条件语句>|<while语句>

<赋值语句> → <变量> := <算术表达式>;

<算术表达式> → <项><算术表达式公因式>

<算术表达式公因式> → ε |**+**<算术表达式>

<项> → <因式><项公因式>

<项公因式> → ε | \*<项> | /<项>

<条件语句> → IF <关系表达式> THEN <语句><条件语句公因式>

<条件语句公因式> → ε | ELSE<语句>

<关系表达式> → <算数表达式> <关系符> <算数表达式>

<while语句> → WHILE <关系表达式> DO <语句表>END;

文法无直接左递归，非终结符排序为：

<程序>，<变量说明>，<变量说明序列>，<变量说明序列公因式>，<变量表>，<变量表公因式> ，<语句表>，<语句公因式>，<语句>，<赋值语句>，<算术表达式>，<项>，<项公因式>，<条件语句>，<关系表达式>，<while语句>

多次推导后无左递归式子，文法判断无左递归

通过构建公因式产生式，已将最左公因式已全部消除

其中设F1为first集，F2位follow集，可得

F1（程序）= { Var } Follow(程序) = {#}

F1（变量说明）= { Var } Follow(变量说明) = {#，Begin}

F1（变量说明序列）= { ID，ε } Follow(变量说明序列) = {#，Begin }

F1（变量说明序列公因式）= { ID，ε } Follow(变量说明序列公因式) = {#，Begin }

F1（变量表）= { ID } Follow(变量表) = {#，: }

F1（语句表）= { 变量，IF，WHILE，ε } Follow(语句表) = {#，END}

F1（语句公因式）= { 变量，IF，WHILE，ε } Follow(语句公因式) = {#，END}

F1（语句）= { 变量，IF，WHILE } Follow(语句) = {#，END }

F1（赋值语句）= { 变量 } Follow(赋值语句) = {#，END }

F1（条件语句）= { IF } Follow(条件语句) = {#，END }

F1（while语句）= { WHILE } Follow(while语句) = {#，END }

F1（条件语句公因式）= { ε，ELSE } Follow(条件语句公因式) = {#，END }

F1（算数表达式）= { 因式 } Follow(算数表达式) = {#，关系符，THEN }

F1（项）= { 因式 } Follow(项) = {#，关系符 }

F1（项公因式）= { ε，\*，/ } Follow(项公因式) = {#，关系符 }

F1（算数表达式公因子）= { ε，+ } Follow(算数表达式公因子) = {#，关系符，THEN }

F1（关系表达式）= { 因式 } Follow(关系表达式) = {#，THEN }

由LL（1）文法的条件可知

对文法G的句子进行确定的自顶向下语法分析的充分必要条件是，G的任意两个具有相同左部的

产生式A—>α|β 满足下列条件：

（1）如果α、β均不能推导出ε，则 FIRST(α) ∩ FIRST(β) = ∅。

（2）α 和 β 至多有一个能推导出 ε。

（3）如果 β \*═> ε，则 FIRST(α) ∩ FOLLOW(A) = ∅。

将满足上述条件的文法称为LL(1)文法。

除显然可见的满足条件，有

由变量说明序列满足（3），变量说明序列公因式满足（3），

语句表满足（3），语句公因式满足（3）

语句满足（1），（3），算术表达式公因子满足（3）

项公因式满足（3），条件语句公因子满足（3）

可知，该文法为LL（1）文法，可进行递归下降分析。