```
1    <prog>         ::= PROGRAM <prog-name> VAR <dec-list> BEGIN <stmt-list> END.
2    <prog-name>    ::= id
3a   <dec-list>     ::= <dec> { ; <dec> }
4    <dec>          ::= <id-list> : <type>
5    <type>         ::= INTEGER
6a   <id-list>      ::= id { , id }
7a   <stmt-list>    ::= <stmt> { ; <stmt> }
8    <stmt>         ::= <assign> | <read> | <write> | <for>
9    <assign>       ::= id := <exp>
10a  <exp>          ::= <term> { + <term> | - <term> }
11a  <term>         ::= <factor> { * <factor> | DIV < factor> }
12   <factor>       ::= id | int | ( <exp> )
13   <read>         ::= READ ( <id-list> )
14   <write>        ::= WRITE ( <id-list> )
15   <for>          ::= FOR <index-exp> DO <body>
16   <index-exp>    ::= id := <exp> TO <exp>
17   <body>         ::= <stmt> | BEGIN <stmt-list> END
```

Figure 5.15 Simplified Pascal grammar modified for recursive-descent parse.