```
 ⟨program⟩  ::=  ⟨block⟩
   ⟨block⟩  ::=  { ⟨decls⟩ ⟨stmts⟩ }
   ⟨decls⟩  ::=  ⟨decls⟩ ⟨decl⟩   |  ε
    ⟨decl⟩  ::=  ⟨type⟩ ID ;
    ⟨type⟩  ::=  ⟨type⟩ [ NUM ]   |  BASIC
   ⟨stmts⟩  ::=  ⟨stmts⟩ ⟨stmt⟩   |  ε
    ⟨stmt⟩  ::=  ⟨loc⟩ = ⟨bool⟩ ;
             |   IF ( ⟨bool⟩ ) ⟨stmt⟩
             |   IF ( ⟨bool⟩ ) ⟨stmt⟩ ELSE ⟨stmt⟩
             |   WHILE ( ⟨bool⟩ ) ⟨stmt⟩
             |   ⟨block⟩
     ⟨loc⟩  ::=  ⟨loc⟩ [ ⟨bool⟩ ]   |  ID
    ⟨bool⟩  ::=  ⟨bool⟩ || ⟨join⟩   |  ⟨join⟩
    ⟨join⟩  ::=  ⟨join⟩ && ⟨equality⟩   |  ⟨equality⟩
⟨equality⟩  ::=  ⟨equality⟩ == ⟨rel⟩
             |   ⟨equality⟩ != ⟨rel⟩
             |   ⟨rel⟩
     ⟨rel⟩  ::=  ⟨expr⟩ < ⟨expr⟩
             |   ⟨expr⟩ <= ⟨expr⟩
             |   ⟨expr⟩ >= ⟨expr⟩
             |   ⟨expr⟩ > ⟨expr⟩
             |   ⟨expr⟩
    ⟨expr⟩  ::=  ⟨expr⟩ + ⟨term⟩
             |   ⟨expr⟩ - ⟨term⟩
             |   ⟨term⟩
    ⟨term⟩  ::=  ⟨term⟩ * ⟨unary⟩
             |   ⟨term⟩ / ⟨unary⟩
             |   ⟨unary⟩
   ⟨unary⟩  ::=  ! ⟨unary⟩
             |   - ⟨unary⟩
             |   ⟨factor⟩
  ⟨factor⟩  ::=  ( ⟨bool⟩ )
             |   ⟨loc⟩
             |   NUM
             |   REAL
             |   TRUE
             |   FALSE
```

Figure 1: A grammar for a simple programming language.

```
    ⟨program⟩  ::=  ⟨block⟩
      ⟨block⟩  ::=  { ⟨decls⟩ ⟨stmts⟩ }
      ⟨decls⟩  ::=  ε   |   ⟨decl⟩ ⟨decls⟩
       ⟨decl⟩  ::=  ⟨type⟩ ID ;
       ⟨type⟩  ::=  BASIC ⟨typecl⟩
     ⟨typecl⟩  ::=  ε   |   [ NUM ] ⟨typecl⟩
      ⟨stmts⟩  ::=  ε   |   ⟨stmt⟩ ⟨stmts⟩
       ⟨stmt⟩  ::=  ⟨loc⟩ = ⟨bool⟩ ; | IF ( ⟨bool⟩ ) ⟨stmt⟩
                |   IF ( ⟨bool⟩ ) ⟨stmt⟩ ELSE ⟨stmt⟩
                |   WHILE ( ⟨bool⟩ ) ⟨stmt⟩
                |   ⟨block⟩
        ⟨loc⟩  ::=  ID ⟨loccl⟩
      ⟨loccl⟩  ::=  ε   |   [ ⟨bool⟩ ] ⟨loccl⟩
       ⟨bool⟩  ::=  ⟨bool⟩ || ⟨join⟩   |   ⟨join⟩
       ⟨join⟩  ::=  ⟨equality⟩ ⟨joincl⟩
     ⟨joincl⟩  ::=  ε   |   && ⟨equality⟩ ⟨joincl⟩
   ⟨equality⟩  ::=  ⟨rel⟩ ⟨equalitycl⟩
 ⟨equalitycl⟩  ::=  ε
                |   == ⟨rel⟩ ⟨equalitycl⟩
                |   != ⟨rel⟩ ⟨equalitycl⟩
        ⟨rel⟩  ::=  ⟨expr⟩ ⟨reltail⟩
    ⟨reltail⟩  ::=  ε   |   < ⟨expr⟩
                |   <= ⟨expr⟩
                |   >= ⟨expr⟩
                |   > ⟨expr⟩
       ⟨expr⟩  ::=  ⟨expr⟩ ⟨termcl⟩
      ⟨expcl⟩  ::=  ε
                |   + ⟨term⟩ ⟨exprcl⟩
                |   - ⟨term⟩ ⟨expel⟩
       ⟨term⟩  ::=  ⟨unary⟩ ⟨termcl⟩
     ⟨termcl⟩  ::=  ε
                |   * ⟨unary⟩ ⟨termcl⟩
                |   / ⟨unary⟩ ⟨termcl⟩
      ⟨unary⟩  ::=  ! ⟨unary⟩
                |   - ⟨unary⟩
                |   ⟨factor⟩
     ⟨factor⟩  ::=  ( ⟨bool⟩ )
                |   ⟨loc⟩
                |   NUM
                |   REAL
                |   TRUE
                |   FALSE
```

Figure 2: A modified grammar suitable for recursive descent parsing.