```
<prog> ::= PROGRAM <prog-name> VAR <dec-list> BEGIN <stmt-list> END.

                generate [  LDL  RETADR]
                generate [  RSUB]
                for each Ti variable used do
                    generate [Ti    RESW    1]
                insert [  J EXADDR] {jump to first executable instruction}
                    in bytes 3-5 of object program
                fix up forward references to Ti variables
                generate Modification records for external references
                generate [  END ]


<prog-name> ::= id

                generate [  START 0]
                generate [  EXTREF  XREAD,XWRITE]
                generate [  STL  RETADR]
                add 3 to LOCCTR{leave room for jump to first executable instruction}
                generate [RETADR    RESW    1]


<dec-list> ::= {either alternative}

                save LOCCTR as EXADDR  {tentative address of first executable instruction}


<dec> ::= <id-list> : <type>

                for each item on list do
                    begin
                        remove S(NAME) from list
                        enter LOCCTR into symbol table as address for NAME
                        generate [S(NAME)   RESW    1]
                    end
                LISTCOUNT := 0


<type> ::= INTEGER

                {no code-generation action}


<stmt-list> ::= {either alternative}

                {no code-generation action}
```

Figure 5.20 Other code-generation routines for the grammar from Fig. 5.2.