After the initial checks to catch subscripted constants and ( immediately following a closing bracket DECTYP is tested to see whether we are in array declaration or in a statement. The state variable M will tell us whether this delimiter is being used as an expression bracket or in a procedure call.

In the case of an array declaration, a failure is indicated if E is set to statement level, unless PROC shows that an actual parameter is being processed, e.g.

real array A [1: PROC (a, PROC2(....

or real array A [1: PROC (a,(b + c ....

The former example will cause M to be set at 1, and the latter zero, which is the subject of the next test. Where this is a procedure call, the flowchart joins with the path where DECTYP was zero and this was a procedure call (which has already called EXP to change E to expression level if necessary, and set EXPRES in the case of a read or print statement e.g. read reader (1)...)

The case of "go to S (..." is failed by the next test on EXPTYP; and the procedure name is then searched for in the Name List. If we are in an array declaration, we check that the array bounds are not local (illegal ALGOL), before discovering whether this is a type procedure. If so, space is reserved on the run-time stack (by compiling PRIM UP) for the result of the type procedure. Various variables are then stacked with the delimiter, and these variables are then set up to deal with the actual parameters of this procedure call.

In the case of expression brackets used in statements, the delimiter is stacked, and PROC set to zero while processing the constituents of the bracket.

ERRORS

| | | |
|---|---|---|
| FAIL 61 | ; | (misplaced |
| FAIL 82 | ; | ( must not appear in a type declaration |
| FAIL 62 | ; | function designator as subject of "go to" |
| FAIL 41 | ; | array bounds must not be local |
| FAIL 25 | ; | non type procedure as function designator in expression |