Consider the program on the right. Change the two assignments to the
LIST array to:

LIST[1] := 3
LIST[2] := 1

Hand execute the new program under the following assumptions and compare
the resulting values in the array LIST in BIGSUB after the return from SUB:
a. parameters are passed by value
b. parameters are passed by reference
c. parameters are passed by name
d. parameters are passed by value-result

Some rather clever tricks can be played with pass-by-name parameters.
The most famous example of this was proposed by J. Jensen of the
Regnecentralen in Copenhagen in 1960 and has since been called Jensen’s
Device. Jensen’s idea involves passing an expression and one or more vari-
ables that appear in that expression as parameters to a subprogram. When-
ever one of the variable form parameters is changed in the subprogram, that
change can cause a change of the values of later occurrences of the formal
parameter that corresponds to the expression actual parameter. Consider
the following ALGOL 60 procedure:

real procedure SUM (ADDER, INDEX, LENGTH);
  value LENGTH;
  real ADDER;
  integer INDEX, LENGTH;
begin
  real TEMPSUM;
  TEMPSUM := 0.0;
  for INDEX := 1 step 1 until LENGTH do
    TEMPSUM := TEMPSUM + ADDER;
    SUM := TEMPSUM
end;

Still another way to get aliasing with pass-by-reference parameters is
through collisions between formal parameters and nonlocal variables that are
visible.

These aliases are possible when a language provides more nonlocal
access than is necessary, such as with static scoping. For example, consider
the following Pascal code:

procedure bigsub;
  var global : integer;
procedure smallsub (var local : integer);
begin
  ...
end; { smallsub }
begin
  ...
smallsub (global);
...
end; { bigsub }

Inside smallsub, local and global are aliases. As stated above, the main
reason for the aliasing is that static scoping provides too much access to
nonlocal variables. The problem with these kinds of aliasing is the same as
in other circumstances: It is harmful to readability and thus to reliability. It
also makes program verification extremely difficult.