ALGORITHM 71
PERMUTATION
R. R. Coveyou and J. G. Sullivan
Oak Ridge National Laboratory, Oak Ridge, Tenn.

procedure PERMUTATION (I, P, N);
  value I, N; integer N; integer array P; boolean I;
  comment This procedure produces all permutations of the
  integers from 0 thru N. Upon entry with I = false the
  procedure initializes itself producing no permutation. Upon each
  successive entry into the procedure with I = true a new
  permutation is stored in P[0] thru P[N]. When the process has
  been exhausted a sentinel is set:
  P[0] := -1;
  N := 0;
begin
  integer i; own integer array x[0:N];
  if not I then
    begin
      for i := 0 step 1 until N-1 do x[i] := 0;
      x[N] := -1;
      go to E end;
    for i := N step -1 until 0 do
      if x[i] = i then go to A;
    x[i] := 0 end;
    P[0] := -1; go to E;
  A: x[i] := x[i]+1; P[i] := 0;
    for i := 1 step 1 until N do
      begin
        P[i] := P[i]-x[i]; P[i]-x[i] := i end;
  E: end PERMUTATION

CERTIFICATION OF ALGORITHM 71
PERMUTATION (R. R. Coveyou and J. G. Sullivan,
Comm. ACM, Nov. 1961)
P. J. Brown
University of North Carolina, Chapel Hill, N. C.

PERMUTATION was transcribed into GAT for the UNI-
VAC 1105 and successfully run for a number of cases.

CERTIFICATION OF ALGORITHM 71
PERMUTATION (R. R. Coveyou and J. G. Sullivan,
Comm. ACM, Nov. 1961)
J. E. L. Peck and G. F. Schrack
University of Alberta, Calgary, Alberta, Canada

PERMUTATION was translated into FORTRAN for the IBM
1620 and it performed satisfactorily. The own integer array
x[0:n] may be shortened to x[1:n], provided corresponding cor-
rections are made in the first two for statements.

However, PERMUTE (Algorithm 86) is superior to PERMU-
TATION in two respects:
(1) PERMUTATION, using storage of order 2n, is designed to
permute the specific vector 0, 1, 2, · · · , n − 1 rather than an
arbitrary vector. Thus storage of order 3n is required to permute
an arbitrary vector. PERMUTE, in contrast, only needs storage
of order 2n to permute an arbitrary vector.
(2) PERMUTE is built up from cyclic permutations. The
number of permutations actually executed internally (the re-
dundant ones are suppressed) by PERMUTE is asymptotic to
(e − 1)n! rather than n!. In spite of this, PERMUTE is dis-
tinctly faster (1316 against 2528 seconds for n = 8) than PERMU-
TATION. If tₙ is the time taken for all permutations of a vector
with n components, and if rₙ = tₙ/tn₋₁, then one would expect
rₙ to be close to 1. Experiment with small values of n gave the
following results for rₙ:

<table>
<thead>
<tr>
<th>n</th>
<th>6</th>
<th>7</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>PERMUTE</td>
<td>0.96</td>
<td>0.99</td>
<td>1.00</td>
</tr>
<tr>
<td>PERMUTATION</td>
<td>1.10</td>
<td>1.13</td>
<td>1.12</td>
</tr>
</tbody>
</table>

Is there yet a faster way to do it?
See also: C. Tompkins, "Machine Attacks on Problems whose
Variables are Permutations", Proceedings of Symposia in Applied
Mathematics, Vol. VI: Numerical Analysis (N. Y., McGraw-Hill,
1956).

CERTIFICATION OF ALGORITHM 71
PERMUTATION [R. R. Coveyou and J. G. Sullivan,
Comm. ACM, Nov. 1961]
J. S. Hillmore

The algorithm was successfully run using the Elliott ALGOL
translator on the National-Elliott 803. The integer array x was
made a parameter of the procedure in order to avoid having an
own array with variable bounds.