```
! Generating all permutations on N items in lexico increasing order;
! We do not use the Heap method, but rather generate in lexicographic order;
!  Ref:
 Heap, B. R. (1963). "Permutations by Interchanges". The Computer Journal. 6 (3): 293-4..
 Sedgewick, R. (1977). "Permutation Generation Methods".
      ACM Computing Surveys. 9(2):137-164.;
!Keywords: Enumerate, Generate, Heap, Lexicographic, LINGO, Ordering, Permutation,
Sedgewick;
SETS:
 ITEM: INPOS;
ENDSETS
DATA:
  ITEM = 1..4; ! Number of items to permute;
ENDDATA

PROCEDURE WRITEOUT:
 ! Write current permuation;
 @WRITE( @FORMAT( iter,'6.0f'),') ');
 @FOR( ITEM( ii):
   @WRITE( INPOS(ii),' ');
     );
 @WRITE( @NEWLINE(1));
ENDPROCEDURE

PROCEDURE GetNxtPrm:
! Inputs:
    NT = number of items in permutation, 1, 2, ..., NT,
    Cprm = 1 on first call,
  Outputs:
   InPos() = next permuation,
   Cprm = 0 if no more permutations
;
! Is this the first call? ;
  @IFC( Cprm #EQ# 0:
! Initial increasing order;
     @FOR( ITEM( i): INPOS( i) = i);
     Cprm = 1; ! Next call will not be first;
   @ELSE

! Find largest Cprm for which INPOS( Cprm) < INPOS(Cprm+1);
   Cprm = 0; ! Default is, no such Cprm;
   i = NT;
   @WHILE( i #GT# 1:
    i = i - 1;
    @IFC( INPOS(i) #LT# INPOS(i+1):
      Cprm = i;
      i = 0;
        );
       );

  @IFC( Cprm #GT# 0: ! Still more to do? ;
! Find largest r for which INPOS(Cprm) < INPOS(r);
   INPK = INPOS(Cprm);
   i = NT+1;
   @WHILE( i #GT# 0:
    i = i - 1;
    @IFC( INPK #LT# INPOS(i) :
      r = i;
      i = 0;
        );
       );

! Swap INPOS(Cprm) and INPOS(r);
  INPOS(Cprm) = INPOS(r);
  INPOS(r) = INPK;

! Reverse the sequence INPOS(Cprm+1:NT);
 i = Cprm;
 j = NT;
 @WHILE( i #LT# j:
   i = i + 1;
   ITMP = INPOS(i);
   INPOS( i) = INPOS( j);
   INPOS( j) = ITMP;
   j = j - 1;
       );
  );
 );
ENDPROCEDURE

CALC:
 ! At each iteration, INPOS(i) will give the item/integer
   in position i at the current iteration's permutation;
 @SET('TERSEO',  2); !Output level (0:verbose, 1:terse, 2:errors only, 3:no output).
Default:0;
 NT = @SIZE(ITEM); ! Number of items in the permutation;
 iter = 0;
 Cprm = 0; ! Set to 0 for first call. All done when back to 0;
 More = 1; ! Keep going until More = 0;
 @WHILE( More:
   iter = iter + 1;
   GetNXTPRM;  ! Get next permutation;
   @IFC( Cprm  #GT# 0: WRITEOUT); ! Write out current permutation;
   More = Cprm; ! Check if last one;
     );
ENDCALC
```