Example: FORTRAN

```
C FORTRAN (OLD-STYLE) SORTING ROUTINE
C
      SUBROUTINE SORT (A, N)
      DIMENSION A(N)
      IF (N - 1) 40, 40, 10
10    DO 30 I = 2, N
         L = I-1
         X = A(I)
         DO 20 J = 1, L
            K = I - J
            IF (X - A(K)) 60, 50, 50
C FOUND INSERTION POINT: X >= A(K)
50             A(K+1) = X
               GO TO 30
C ELSE, MOVE ELEMENT UP
60             A(K+1) = A(K)
20       CONTINUE
         A(1) = X
30    CONTINUE
40    RETURN
      END
```

```
C --------------------------------

C MAIN PROGRAM
      DIMENSION Q(500)
100   FORMAT(I5/(6F10.5))
200   FORMAT(6F12.5)

      READ(5, 100) N, (Q(J), J = 1, N)
      CALL SORT(Q, N)
      WRITE(6, 200) (Q(J), J = 1, N)
      STOP
      END
```


Example: Algol 60

```
comment An Algol 60 sorting program;
procedure Sort (A, N)
   value N;
   integer N; real array A;
begin
   real X;
   integer i, j;
   for i := 2 until N do begin
      X := A[i];
      for j := i-1 step -1 until 1 do
         if X >= A[j] then begin
            A[j+1] := X; goto Found
         end else
             A[j+1] := A[j];
      A[1] := X;
   Found:
      end
   end
end Sort
```


Example: APL

```
⍝ An APL sorting program
∇ Z ← SORT A
  Z ← A [⍋A]
∇
```


Example: Prolog

```
/* A naive Prolog sort */

/* permutation(A,B) iff list B is a
   permutation of list A. */
permutation(L, [H | T]) :-
   append(V,[H|U],L),
   append(V,U,W),
   permutation(W,T).
permutation([], []).

/* ordered(A) iff A is in ascending order. */
ordered([]).
ordered([X]).
ordered([X,Y|R]) :- X <= Y, ordered([Y|R]).

/* sorted(A,B) iff B is a sort of A. */
sorted(A,B) :- permutation(A,B), ordered(B).
```