Insertion Sort

Pseudocode of the complete algorithm follows, where the arrays are zero-based:

```
for i ← 1 to length(A) - 1
    j ← i
    while j > 0 and A[j-1] > A[j]
        swap A[j] and A[j-1]
        j ← j – 1
```

The outer loop runs over all the elements except the first one, because the single-element prefix `A[0:1]` is trivially sorted, so the invariant that the first `i+1` entries are sorted is true from the start. The inner loop moves element `A[i]` to its correct place so that after the loop, the first `i+2` elements are sorted.
After expanding the "swap" operation in-place as `t ← A[j]; A[j] ← A[j-1]; A[j-1] ← t` (where `t` is a temporary variable), a slightly faster version can be produced that moves `A[i]` to its position in one go and only performs one assignment in the inner loop body:

```
for i = 1 to length(A) - 1
    x = A[i]
    j = i
    while j > 0 and A[j-1] > x
        A[j] = A[j-1]
        j = j - 1
    A[j] = x
```

The new inner loop shifts elements to the right to clear a spot for `x = A[i]`.
Note that although the common practice is to implement in-place, which requires checking the elements in-order, the order of checking (and removing) input elements is actually arbitrary. The choice can be made using almost any pattern, as long as all input elements are eventually checked (and removed from the input).