order. But the subarray $A[1 \ldots n]$ is the entire array! Hence, the entire array is sorted, which means that the algorithm is correct.

We shall use this method of loop invariants to show correctness later in this chapter and in other chapters as well.

Pseudocode conventions

We use the following conventions in our pseudocode.

1. Indentation indicates block structure. For example, the body of the **for** loop that begins on line 1 consists of lines 2–8, and the body of the **while** loop that begins on line 5 contains lines 6–7 but not line 8. Our indentation style applies to **if**-**then**-**else** statements as well. Using indentation instead of conventional indicators of block structure, such as **begin** and **end** statements, greatly reduces clutter while preserving, or even enhancing, clarity.[2]
2. The looping constructs **while**, **for**, and **repeat** and the conditional constructs **if**, **then**, and **else** have interpretations similar to those in Pascal.[3] There is one subtle difference with respect to **for** loops, however: in Pascal, the value of the loop-counter variable is undefined upon exiting the loop, but in this book, the loop counter retains its value after exiting the loop. Thus, immediately after a **for** loop, the loop counter's value is the value that first exceeded the **for** loop bound. We used this property in our correctness argument for insertion sort. The **for** loop header in line 1 is **for** $j \leftarrow 2$ **to** $length[A]$, and so when this loop terminates, $j = length[A]+1$ (or, equivalently, $j = n+1$, since $n = length[A]$).
3. The symbol "$\triangleright$" indicates that the remainder of the line is a comment.
4. A multiple assignment of the form $i \leftarrow j \leftarrow e$ assigns to both variables i and j the value of expression e; it should be treated as equivalent to the assignment $j \leftarrow e$ followed by the assignment $i \leftarrow j$.
5. Variables (such as i, j, and key) are local to the given procedure. We shall not use global variables without explicit indication.
6. Array elements are accessed by specifying the array name followed by the index in square brackets. For example, $A[i]$ indicates the ith element of the array A. The notation ". ." is used to indicate a range of values within an ar-

---

[2] In real programming languages, it is generally not advisable to use indentation alone to indicate block structure, since levels of indentation are hard to determine when code is split across pages.

[3] Most block-structured languages have equivalent constructs, though the exact syntax may differ from that of Pascal.

ray. Thus, $A[1 \mathinner{..} j]$ indicates the subarray of A consisting of the j elements $A[1], A[2], \ldots, A[j]$.

7. Compound data are typically organized into ***objects***, which are composed of ***attributes*** or ***fields***. A particular field is accessed using the field name followed by the name of its object in square brackets. For example, we treat an array as an object with the attribute *length* indicating how many elements it contains. To specify the number of elements in an array A, we write $length[A]$. Although we use square brackets for both array indexing and object attributes, it will usually be clear from the context which interpretation is intended.

   A variable representing an array or object is treated as a pointer to the data representing the array or object. For all fields f of an object x, setting $y \leftarrow x$ causes $f[y] = f[x]$. Moreover, if we now set $f[x] \leftarrow 3$, then afterward not only is $f[x] = 3$, but $f[y] = 3$ as well. In other words, x and y point to the same object after the assignment $y \leftarrow x$.

   Sometimes, a pointer will refer to no object at all. In this case, we give it the special value NIL.

8. Parameters are passed to a procedure ***by value***: the called procedure receives its own copy of the parameters, and if it assigns a value to a parameter, the change is *not* seen by the calling procedure. When objects are passed, the pointer to the data representing the object is copied, but the object's fields are not. For example, if x is a parameter of a called procedure, the assignment $x \leftarrow y$ within the called procedure is not visible to the calling procedure. The assignment $f[x] \leftarrow 3$, however, is visible.

9. The boolean operators "and" and "or" are ***short circuiting***. That is, when we evaluate the expression "x and y" we first evaluate x. If x evaluates to FALSE, then the entire expression cannot evaluate to TRUE, and so we do not evaluate y. If, on the other hand, x evaluates to TRUE, we must evaluate y to determine the value of the entire expression. Similarly, in the expression "x or y" we evaluate the expression y only if x evaluates to FALSE. Short-circuiting operators allow us to write boolean expressions such as "$x \neq$ NIL and $f[x] = y$" without worrying about what happens when we try to evaluate $f[x]$ when x is NIL.

Exercises

2.1-1

Using Figure 2.2 as a model, illustrate the operation of INSERTION-SORT on the array $A = \langle 31, 41, 59, 26, 41, 58 \rangle$.