



**Figure 8.2** The operation of COUNTING-SORT on an input array  $A[1..8]$ , where each element of  $A$  is a nonnegative integer no larger than  $k = 5$ . (a) The array  $A$  and the auxiliary array  $C$  after line 5. (b) The array  $C$  after line 8. (c)–(e) The output array  $B$  and the auxiliary array  $C$  after one, two, and three iterations of the loop in lines 10–12, respectively. Only the lightly shaded elements of array  $B$  have been filled in. (f) The final sorted output array  $B$ .

COUNTING-SORT( $A, B, k$ )

```

1  let  $C[0..k]$  be a new array
2  for  $i = 0$  to  $k$ 
3       $C[i] = 0$ 
4  for  $j = 1$  to  $A.length$ 
5       $C[A[j]] = C[A[j]] + 1$ 
6  //  $C[i]$  now contains the number of elements equal to  $i$ .
7  for  $i = 1$  to  $k$ 
8       $C[i] = C[i] + C[i - 1]$ 
9  //  $C[i]$  now contains the number of elements less than or equal to  $i$ .
10 for  $j = A.length$  downto 1
11      $B[C[A[j]]] = A[j]$ 
12      $C[A[j]] = C[A[j]] - 1$ 

```

Figure 8.2 illustrates counting sort. After the **for** loop of lines 2–3 initializes the array  $C$  to all zeros, the **for** loop of lines 4–5 inspects each input element. If the value of an input element is  $i$ , we increment  $C[i]$ . Thus, after line 5,  $C[i]$  holds the number of input elements equal to  $i$  for each integer  $i = 0, 1, \dots, k$ . Lines 7–8 determine for each  $i = 0, 1, \dots, k$  how many input elements are less than or equal to  $i$  by keeping a running sum of the array  $C$ .