```
clear1(int array[], int size)
{
    int i;
    for (i = 0; i < size; i += 1)
        array[i] = 0;
}
```

```
clear2(int *array, int size)
{
    int *p;
    for (p = &array[0]; p < &array[size]; p = p + 1)
        *p = 0;
}
```

FIGURE 2.30 Two C procedures for setting an array to all zeros. Clear1 uses indices, while clear2 uses pointers. The second procedure needs some explanation for those unfamiliar with C. The address of a variable is indicated by &, and the object pointed to by a pointer is indicated by *. The declarations declare that array and p are pointers to integers. The first part of the *for* loop in clear2 assigns the address of the first element of array to the pointer p. The second part of the for loop tests to see if the pointer is pointing beyond the last element of array. Incrementing a pointer by one, in the last part of the *for* loop, means moving the pointer to the next sequential object of its declared size. Since p is a pointer to integers, the compiler will generate MIPS instructions to increment p by four, the number of bytes in a MIPS integer. The assignment in the loop places 0 in the object pointed to by p.