```
{"Indonesia",         62}, {"Iran",            98},
{"Italy",             39}, {"Japan",           81},
{"Mexico",            52}, {"Nigeria",        234},
{"Pakistan",          92}, {"Philippines",     63},
{"Poland",            48}, {"Russia",           7},
{"South Africa",      27}, {"South Korea",     82},
{"Spain",             34}, {"Sudan",          249},
{"Thailand",          66}, {"Turkey",          90},
{"Ukraine",          380}, {"United Kingdom",  44},
{"United States",      1}, {"Vietnam",         84}};
```

The inner braces around each structure value are optional. As a matter of style, however, I prefer not to omit them.

C99 Because arrays of structures (and structures containing arrays) are so common, C99's designated initializers allow an item to have more than one designator. Suppose that we want to initialize the inventory array to contain a single part. The part number is 528 and the quantity on hand is 10, but the name is to be left empty for now:

```
struct part inventory[100] =
  {[0].number = 528, [0].on_hand = 10, [0].name[0] = '\0'};
```

The first two items in the list use two designators (one to select array element 0—a part structure—and one to select a member within the structure). The last item uses three designators: one to select an array element, one to select the name member within that element, and one to select element 0 of name.

PROGRAM Maintaining a Parts Database

To illustrate how nested arrays and structures are used in practice, we'll now develop a fairly long program that maintains a database of information about parts stored in a warehouse. The program is built around an array of structures, with each structure containing information—part number, name, and quantity—about one part. Our program will support the following operations:

- *Add a new part number, part name, and initial quantity on hand.* The program must print an error message if the part is already in the database or if the database is full.
- *Given a part number, print the name of the part and the current quantity on hand.* The program must print an error message if the part number isn't in the database.
- *Given a part number, change the quantity on hand.* The program must print an error message if the part number isn't in the database.
- *Print a table showing all information in the database.* Parts must be displayed in the order in which they were entered.
- *Terminate program execution.*