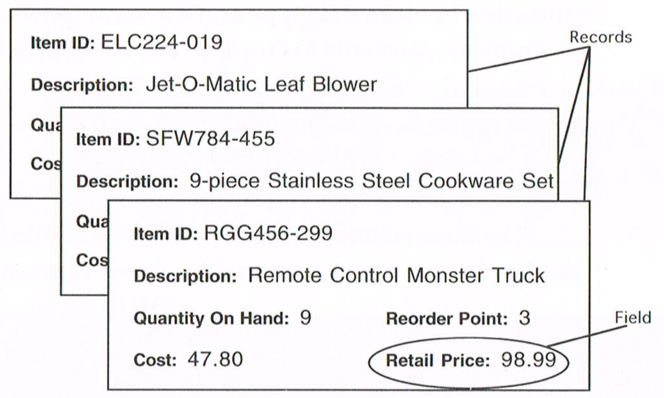

FIGURE 10-7
A record in a database is one completed set of fields.

Note

All data in a C++ program is stored in a ***data structure****. Any organized way of storing data in a computer is a data structure. The basic variable types such as* `int` *and* `float` *are called* ***primitive data structures****. A character array is an example of a category of data structures called* ***simple data structures****. The term structure used in this section refers to a specific data structure made by grouping other data structures. Do not confuse the term structure used in this section with the more generic term data structure.*

In a database program, data is stored in ***records***. For example, suppose you have a database of items sold by a mail-order company. Each item that the company sells is stored as a record in the database. Each record is made up of data called ***fields***. Figure 10-7 shows a series of three records contained in a database. Notice that each record has identical field names (i.e., Item ID).

C++ allows you to create a record by grouping the variables and arrays necessary for the fields into a single structure. The variables in the structure can be of mixed types. For example, in Figure 10-7, character arrays must be used for the item ID and the description, an integer type can be used to store the quantity on hand and reorder point, and a floating-point type is necessary for cost and retail price.

DECLARING AND USING STRUCTURES

A structure must be declared. Because a structure is made up of more than one variable, a special syntax is used to access the individual variables of a structure.

DECLARING A STRUCTURE

Shown below is the declaration for the structure in our example.

```
struct inventory_item
  {
    char item_ID[11];
    char description[31];
    int quantity_on_hand;
    int reorder_point;
    float cost;
```

```
    float retail_price;
  };
```

The `struct` keyword identifies the declaration as a structure. The identifier associated with the structure is `inventory_item`. The variables in the structure are called ***members***. The members of the structure are placed within braces. Within the braces, however, the variables and arrays are declared using the syntax to which you are accustomed.

Once you have declared a structure, you must declare a variable that is of the structure's type. This may seem confusing, but what the struct key word does is define a new data type. You can then create as many variables as you want of the new type. The statement below creates a variable named `todays_special` that is of type `inventory_item`.

```
inventory_item todays_special;
```

ACCESSING MEMBERS OF A STRUCTURE

Accessing data in a structure is surprisingly simple. To access a member of the structure, use the name of the variable, a period (.), then the name of the member you need to access, as shown in Figure 10-8. The period is actually an operator called the ***dot operator***.

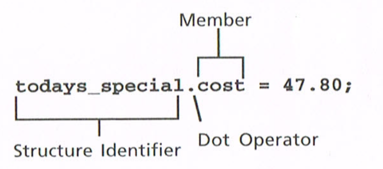

FIGURE 10-8
To access a member of a structure, use the name of the variable, a period, and the name of the member you need to access.

The code segment below declares a variable named `todays_special` of the type `inventory_item` and initializes each member of the structure.

```
inventory_item todays_special;

strcpy(todays_special.item_ID, "RGG456-299");
strcpy(todays_special.description, "Remote Control Monster Truck");
todays_special.quantity_on_hand = 19;
todays_special.reorder_point = 3;
todays_special.cost = 47.80;
todays_special.retail_price = 98.99;
```

EXERCISE 10-7 STRUCTURES

1. Retrieve the source code file *STRUCT.CPP*. A program appears that includes the declaration and initialization of the `todays_special` structure variable.
2. Enter the following code at the bottom of the program (before the closing brace, of course).

```
cout << "Today's Special\n";
cout << "        Item ID: " << todays_special.item_ID << endl;
```