CASE STUDY

# 9 Demetris Leadership Center Case Study—Part 1

The Demetris Leadership Center (DLC, Inc.) publishes the books, DVDs, and audio CDs listed in Table 9A.

**Table 9A** DLC Product Line

| Product Number | Product Title | Product Description | Unit Price | Units Sold |
|---|---|---|---|---|
| 914 | Six Steps to Leadership | Book | $12.95 | 842 |
| 915 | Six Steps to Leadership | Audio CD | $14.95 | 416 |
| 916 | The Road to Excellence | DVD | $18.95 | 127 |
| 917 | Seven Lessons of Quality | Book | $16.95 | 514 |
| 918 | Seven Lessons of Quality | Audio CD | $21.95 | 437 |
| 919 | Seven Lessons of Quality | DVD | $31.95 | 269 |
| 920 | Teams are Made, Not Born | Book | $14.95 | 97 |
| 921 | Leadership for the Future | Book | $14.95 | 492 |
| 922 | Leadership for the Future | Audio CD | $16.95 | 212 |

The manager of the Telemarketing Group has asked you to write a program that will help order-entry operators look up product prices. The program should prompt the user to enter a product number and then display the title, description, and price of the product.

### Structure

The program will use an array of `ProdStruct` structures to hold the product information. The structure has one member for each field of information to be stored. Table 9B lists the members of the `ProdStruct` structure.

**Table 9B** Members of the `ProdStruct` Structure

| Member | Data type | Description |
|---|---|---|
| `id` | `int` | Product number |
| `title` | `string` | Title |
| `description` | `string` | Description |
| `price` | `double` | Unit price |
| `sold` | `int` | Units sold during the past 6 months |

### Variables

Table 9C lists the variables to be used in the program.

**Table 9C** Variables Used in the DLC Sales Program

| Variable | Data type | Description |
|---|---|---|
| `product` | `ProdStruct` | Array of structures holding the product data. There is one array element for each product carried by DLC. |
| `NUM_PRODS` | `const int` | Number of products carried by DLC |
| `MIN_PROD_NUM` | `const int` | Lowest valid product number |
| `MAX_PROD_NUM` | `const int` | Highest valid product number |
| `prodNum` | `int` | Product number input by the user |
| `index` | `int` | Array index used to hold the location of a record |
| `again` | `char` | Holds user's choice (y/n) to do another search |

### Modules

The program will consist of the functions listed in Table 9D.

**Table 9D** Functions in the DLC Sales Program

| Function | Description |
|---|---|
| `main` | The program's main function. It calls the program's other functions. |
| `getProdNumber` | This function accepts, validates, and returns a product number input by the user. |
| `binarySearch` | A binary search routine modified to search through an array of `ProdStruct` structures searching for a specific value stored in the `id` member of the structure. If the desired `id` value is found, the subscript of the element holding the structure is returned. If the value is not found, –1 is returned. |
| `displayProd` | Displays the product `id`, `title`, `description`, and `price` members of the structure whose array index is passed to the function. |

### Function `main`

Function `main` contains the variable definitions and calls the other functions. Here is its pseudocode:

```
Initialize numProds, minProdNum, and maxProdNum
Set up the product array and initialize it with all the product records
Do
   Call getProdNum            // Returns the id the user wants
   Call binarySearch          // Returns the index of the desired record
   If binarySearch returned –1
      Display a message that product was not found
   Else
      Call displayProd        // Displays the record data
   End If
   Ask the user if the program should search for another record
   Input again
While again equals 'y' or 'Y'
```

### The `getProdNum` Function

The `getProdNum` function prompts the user to enter a product number. It tests the value to ensure it is in the range of 914 to 922 (which are the valid product numbers). If an invalid value is entered, it is rejected and the user is prompted again. When a valid product number is entered, the function returns it. The pseudocode is as follows:

```
Display a prompt to enter a product number
Read prodNum
While prodNum is invalid
    Display an error message
    Read prodNum
End While
Return prodNum
```

### The `binarySearch` Function

The `binarySearch` function is identical to the function discussed earlier in this chapter, with two changes. First, instead of receiving an array of integers as the earlier `binarySearch` function did, the revised search function receives an array of `ProdStruct` structures. Its function header looks like this:

```
int binarySearch(ProdStruct array[], int size, int value)
```

Second, because each array element is now a structure holding an entire set of data fields, the value being searched for can no longer be compared to an entire array element. Instead, as we did with the array of class objects in Program 9-3 of the text, it must be compared to one of the members, or fields, of the structure. Recall that the specific field being searched on is sometimes called the *key field*. In this program, the data passed to the `value` parameter is a product id number. Therefore, the key field for the search is the `id` field. The two lines of code in the earlier `binarySearch` function that compared the `value` parameter to an array element are modified to compare the parameter to the `id` field of an array element, as shown here:

```
if (array[middle].id == value)

else if (array[middle].id > value)
```

### The `displayProd` Function

The `displayProd` function has two parameters, one to receive the `product` array and one to receive the index of the structure within the array whose data members are to be displayed. It displays the `id`, `title`, `description` and `price` fields of this array element.

### The Entire Program

Here is the entire program's source code.

**DLC1.cpp**

```
 1 // Demetris Leadership Center (DLC) Product Lookup Program
 2 // This program manages an array of product structures.
 3 // It allows the user to enter a product number, then finds
 4 // and displays information on that product.
 5 #include <iostream>
 6 #include <string>
 7 using namespace std;
 8
 9 struct ProdStruct
10 {
11     int    id;               // Product number
12     string title,            // Product title
13            description;      // Product description
14     double price;            // Product unit price
15     int    sold;             // Units sold during the past 6 months
16
17     // Default constructor for a ProdStruct structure
18     ProdStruct()
19     {  price = id = sold = 0;
20        title = description = "";
21     }
22     // Constructor to set initial data values
23     ProdStruct(int i, string t, string d, double p, int s)
24     {  id = i;
25        title = t;
26        description = d;
27        price = p;
28        sold = s;
29     }
30  };
31
32 // Function prototypes
33 int getProdNum(int, int);
34 int binarySearch(ProdStruct [], int, int);
35 void displayProd(ProdStruct [], int);
36
```

```
int main()
{
    const int NUM_PRODS = 9,        // Number of products carried by DLC
              MIN_PROD_NUM = 914,   // Lowest product number
              MAX_PROD_NUM = 922;   // Highest product number

    // Create and initialize the array of ProdStruct structures
    ProdStruct product[NUM_PRODS] =
    {
       ProdStruct(914, "Six Steps to Leadership",  "Book",     12.95, 842),
       ProdStruct(915, "Six Steps to Leadership",  "Audio CD", 14.95, 416),
       ProdStruct(916, "The Road to Excellence",   "DVD",      18.95, 127),
       ProdStruct(917, "Seven Lessons of Quality", "Book",     16.95, 514),
       ProdStruct(918, "Seven Lessons of Quality", "Audio CD", 21.95, 437),
       ProdStruct(919, "Seven Lessons of Quality", "DVD",      31.95, 269),
       ProdStruct(920, "Teams are Made, Not Born", "Book",     14.95,  97),
       ProdStruct(921, "Leadership for the Future","Book",     14.95, 492),
        ProdStruct(922, "Leadership for the Future","Audio CD", 16.95, 212)
    };

    int prodNum,     // Product number the user wants
        index;       // Array subscript where the record is found
    char again;      // Does user want to look up another record (y/n)?

    do
    {  // Get the desired product number
       prodNum = getProdNum(MIN_PROD_NUM, MAX_PROD_NUM);

       // Find the array index of the record for that product
       index = binarySearch(product, NUM_PRODS, prodNum);

       if (index == -1)
          cout << "That product number was not found.\n";
       else
          displayProd(product, index);

       cout << "\nWould you like to look up another product? (y/n) ";
       cin  >> again;
    } while (again == 'y' || again == 'Y');
    return 0;
}

/*********************************************************
 *                          getProdNum                   *
 * Passed in: legal minimum and maximum product numbers  *
 * Returned : a valid product number                     *
 *                                                       *
 * The getProdNum function accepts, validates, and returns   *
 * a product number input by the user.                   *
 *********************************************************/
int getProdNum(int min, int max)
{
    int prodNum;

```

```
 91     // Input the desired product number
 92     cout << "Enter the item's product number "
 93          << min << " - " << max << ": ";
 94     cin  >> prodNum;
 95
 96     // Validate input
 97     while (prodNum < min || prodNum > max)
 98     {  cout << "Invalid product number.\n"
 99             << "Enter the item's product number "
100             << min << " - " << max << ": ";
101        cin  >> prodNum;
102     }
103     return prodNum;
104 }
105
106 /*************************************************************
107  *                          binarySearch                     *
108  * Passed in: the product array, its size, and the product   *
109  *            ID being searched for                          *
110  * Returned : the array index for the record with that ID    *
111  *                                                           *
112  * This function is modified to do a binary search on the id *
113  * field of an array of ProdStruct structures, looking for a *
114  * record (i.e., an array element) whose id member matches   *
115  * value, which holds the product id passed to the function. *
116  * If the record is found, its array subscript is returned.  *
117  * If it is not found , -1 is returned.                      *
118  *************************************************************/
119 int binarySearch(ProdStruct array[], int size, int value)
120 {
121     int  first = 0,                  // First array element
122          last = size - 1,            // Last array element
123          middle,                     // Midpoint of search
124          position = -1;              // Position of search value
125     bool found = false;              // Flag
126
127     while (!found && first <= last)
128     {
129        middle = (first + last) / 2;    // Calculate midpoint
130        if (array[middle].id == value)  // If value is found at mid
131        {
132           found = true;
133           position = middle;
134        }
135        else if (array[middle].id > value) // If value is in lower half
136           last = middle - 1;
137        else
138           first = middle + 1;             // If value is in upper half
139     }
140     return position;
141 }
142
```

```
143 /***************************************************************
144  *                          displayProd                        *
145  * Passed in: the product array and the index of a specific    *
146  *            element of that array                            *
147  *                                                             *
148  * This function displays 4 fields (i.e., structure members) of *
149  * the array element whose index is passed to the function.    *
150  ***************************************************************/
151  void displayProd(ProdStruct product[], int index)
152  {
153     cout << "\nID:          "  << product[index].id;
154     cout << "\nTitle:       "  << product[index].title;
155     cout << "\nDescription: "  << product[index].description;
156     cout << "\nPrice:       $" << product[index].price << endl;
157 }
```

**Program Output with Example Input Shown in Bold**

```
Enter the item's product number 914 - 922: 900[Enter]
Invalid product number.
Enter the item's product number 914 - 922: 916[Enter]

ID:          916
Title:       The Road to Excellence
Description: DVD
Price:       $18.95

Would you like to look up another product? (y/n) y[Enter]
Enter the item's product number 914 - 922: 920[Enter]

ID:          920
Title:       Teams are Made, Not Born
Description: Book
Price:       $14.95

Would you like to look up another product? (y/n) n[Enter]
```