CASE STUDY

9 Demetris Leadership Center Case Study—Part 2

This case study develops another program for the Demetris Leadership Center. Recall from the first DLC case study that DLC, Inc. publishes books, DVDs, and audio CDs. (See Table 9A in Part 1 of this case study for a complete list of products, with product number, title, description, price and sales figures.)

The vice president of sales has asked you to write a sales reporting program that displays the following information:

- A list of the products in the order of their sales dollars (not units sold), from highest to lowest
- The total number of all units sold
- The total sales for the six-month period

Structures

In addition to the `ProdStruct` structure, described in Case Study Part 1 Table 9B, the program will need a `SalesStruct` structure to hold the product number and dollar sales amount of a product. Table 9E lists the members of the `SalesStruct` structure.

Table 9E Members of the `SalesStruct` Structure

Member	Data type	Description
`id`	`int`	Product number of a product
`dollarAmt`	`double`	Dollar amount of sales for that product

Variables

Table 9F lists the variables the program will use.

Table 9F Variables Used in the DLC Sales Report Program

Variable	Data type	Description
`NUM_PRODS`	`const int`	Number of products carried by DLC
`Product`	`ProdStruct`	Array of structures holding the product data described in Table 9B. There is one array element for each product carried by DLC.
`sales`	`SalesStruct`	Array of structures holding the sales data described in Table 9E. There is one array element for each product carried by DLC.

Modules

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

Table 9G Functions in the DLC Sales Report Program

Function	Description
`main`	The program's `main` function. It calls the program's other functions.
`calcSales`	Calculates each product's sales.
`sortBySales`	Sorts the `sales` array so the elements are ordered by `dollarAmt` from highest to lowest.
`showOrder`	Displays a list of the product numbers and their dollar sales amounts.
`showTotals`	Displays the total number of units sold and the total sales amount for the period.

Function `main`

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

```
Initialize NUM_PRODS
Set up the product array and
   initialize it with all the product records
Call calcSales
call sortBySales
Call showOrder
Call showTotals
```

The `calcSales` Function

The `calcSales` function multiplies each product's units sold by its price. The resulting amount is stored in the `sales` array. Here is the function's pseudocode:

```
For index = 0 to the last array subscript
   Set sales[index].id to product[index].id
   Set sales[index].dollarAmt to
      product[index].price * product[index].sold
End For
```

The `sortBySales` Function

The `sortBySales` function is a modified version of the selection sort algorithm shown in Program 9-5 in the text. This version of the function accepts an array of `SalesStruct` elements, rather than an array of integers, and it sorts them in descending order based on the value of the `dollarAmt` structure member. Notice that when an array element needs to be moved, the entire structure can be moved together. It is not necessary to move each of its data members individually. Here is the pseudocode for the `sortBySales` function.

```
For startScan = 0 to the next-to-last subscript
   Set maxIndex to startScan
   Set maxValue to sales[startScan]
   For index = (startScan + 1) to the last array subscript
      If sales[index].dollarAmt is greater than maxValue.dollarAmt
         Set maxValue to sales[index]
         Set maxIndex to index
      End If
   End For
   Set sales[maxIndex] to sales[startScan]
   Set sales[startScan] to maxValue
End For
```

The `showOrder` Function

The `showOrder` function displays a heading and the sorted list of product numbers and their sales amounts. Here is its pseudocode.

```
Display heading
For index = 0 to the last array subscript
   Display sales[index].id
   Display sales[index].dollarAmt
End For
```

The `showTotals` Function

The `showTotals` function displays the total number of units of all products sold and the total sales for the period. It accepts the `units` and `sales` arrays as arguments. Here is its pseudocode.

```
Set totalUnits to 0
Set totalSales to 0.0
For index = 0 to the last array subscript
   Add product[index].sold to totalUnits
   Add sales[index].dollarAmt to totalSales
End For
Display totalUnits with appropriate heading
Display totalSales with appropriate heading
```

The Entire Program

Here is the program's source code.

DLC2.cpp

```
// This program produces a sales report for the Demetris
// Leadership Center. It uses an array of structures.
#include <iostream>
#include <iomanip>
#include <string>
using namespace std;

struct ProdStruct
{
   int    id;                 // Product number
   string title,              // Product title
          description;        // Product description
   double price;              // Product unit price
   int    sold;               // Units sold during the past 6 months

    // Default constructor for a ProdStruct structure
   ProdStruct()
   {  price = id = sold = 0;
      title = description = "";
   }

   // Constructor to set initial data values
   ProdStruct(int i, string t, string d, double p, int s)
   {  id = i;
      title = t;
      description = d;
      price = p;
      sold = s;
   }
};

struct SalesStruct
{
   int     id;                // Product number
   double  dollarAmt;         // Dollar amount of sales in past 6 months
};

// Function prototypes
void calcSales(ProdStruct[], SalesStruct [], int);
void sortBySales(SalesStruct [], int);
void showOrder(SalesStruct [], int);
void showTotals(ProdStruct[], SalesStruct [], int);

```

```
44 int main()
45 {
46    const int NUM_PRODS = 9;    // Number of products carried
47    ProdStruct product[NUM_PRODS] =
48    {
49       ProdStruct(914, "Six Steps to Leadership",  "Book",     12.95,  842),
50       ProdStruct(915, "Six Steps to Leadership",  "Audio CD", 14.95,  416),
51       ProdStruct(916, "The Road to Excellence",   "DVD",      18.95,  127),
52       ProdStruct(917, "Seven Lessons of Quality", "Book",     16.95,  514),
53       ProdStruct(918, "Seven Lessons of Quality", "Audio CD", 21.95,  437),
54       ProdStruct(919, "Seven Lessons of Quality", "DVD",      31.95,  269),
55       ProdStruct(920, "Teams are Made, Not Born", "Book",     14.95,   97),
56       ProdStruct(921, "Leadership for the Future","Book",     14.95,  492),
57       ProdStruct(922, "Leadership for the Future","Audio CD", 16.95,  212)
58    };
59    SalesStruct sales[NUM_PRODS];
60
61    calcSales(product, sales, NUM_PRODS);
62    sortBySales(sales, NUM_PRODS);
63    cout << fixed << showpoint << setprecision(2);
64    showOrder(sales, NUM_PRODS);
65    showTotals(product, sales, NUM_PRODS);
66    return 0;
67 }
68
69 /*************************************************************
70  *                       calcSales                           *
71  * Passed in: the product array, the sales array, and the    *
72  *            size of the arrays                             *
73  *                                                           *
74  * This function uses data in the product array to get the   *
75  * product id and to calculate the product dollarAmt to be   *
76  * stored for each product in the sales array.               *
77  *************************************************************/
78 void calcSales(ProdStruct product[], SalesStruct sales[], int numProds)
79 {
80    for (int index = 0; index < numProds; index++)
81    {  sales[index].id = product[index].id;
82       sales[index].dollarAmt = product[index].price * product[index].sold;
83    }
84 }
85
```

```
 86 /***********************************************************
 87  *                         sortBySales                     *
 88  * Passed in: the sales array and its size                 *
 89  *                                                         *
 90  * This function performs a selection sort, arranging array  *
 91  * elements in descending-order based on the value of the    *
 92  * dollarAmt structure member.                             *
 93  ***********************************************************/
 94 void sortBySales(SalesStruct sales[], int size)
 95 {
 96    int startScan, maxIndex;
 97    SalesStruct maxValue;   // Holds structure with largest dollarAmt so far
 98
 99    for (startScan = 0; startScan < (size - 1); startScan++)
100    {
101       maxIndex = startScan;
102       maxValue = sales[startScan];
103       for (int index = startScan + 1; index < size; index++)
104       {
105          if (sales[index].dollarAmt > maxValue.dollarAmt)
106          {
107             maxValue = sales[index];
108             maxIndex = index;
109          }
110       }
111      sales[maxIndex] = sales[startScan];
112      sales[startScan] = maxValue;
113    }
114 }
115
116 /***********************************************************
117  *                        showOrder                        *
118  * Passed in: the sales array and its size                 *
119  *                                                         *
120  * This function displays the product number and dollar sales *
121  * amount of each product DLC sells.                       *
122  ***********************************************************/
123 void showOrder(SalesStruct sales[], int numProds)
124 {
125    cout << "Product ID \t  Sales\n";
126    cout << "---------------------------\n";
127    for (int index = 0; index < numProds; index++)
128    {
129       cout << sales[index].id << "\t\t  $";
130       cout << setw(8) << sales[index].dollarAmt << endl;
131    }
132     cout << endl;
133 }
134
```

```
135 /***********************************************************
136  *                        showTotals                       *
137  * Passed in: the product array, the sales array, and the  *
138  *            size of the arrays                           *
139  *                                                         *
140  * This function calculates and displays the total quantity *
141  * of items sold and the total dollar amount of sales.     *
142  ***********************************************************/
143 void showTotals(ProdStruct product[], SalesStruct sales[], int numProds)
144 {
145    int totalUnits = 0;
146    double totalSales = 0.0;
147
148    for (int index = 0; index < numProds; index++)
149    {
150       totalUnits += product[index].sold;
151       totalSales += sales[index].dollarAmt;
152    }
153    cout << "Total units Sold:  " << totalUnits << endl;
154    cout << "Total sales:      $" << totalSales << endl;
155 }
```

Program Output

```
Product Number     Sales
---------------------------------
914               $10903.90
918               $ 9592.15
917               $ 8712.30
919               $ 8594.55
921               $ 7355.40
915               $ 6219.20
922               $ 3593.40
916               $ 2406.65
920               $ 1450.15

Total Units Sold:  3406
Total Sales:      $58827.70
```