

Laboratory 12B

CS-102

Spring 2022

Inventory File Management System

- Write a program that uses a Struct to store the following data about a Inventory File Management System:
 - ItemName
 - ID_Number
 - Quantity
 - Price
- You will call your program: *YourName*-Lab12B.cpp

Random Files: Opening Files For Both Input and Output

- Enter the following data into each of the five records:

<u>Desc</u>	<u>ID</u>	<u>qty</u>	<u>price</u>
Wrench	1	10	\$4.67
Hammer	2	15	\$3.97
Saw	3	5	\$4.95
Pliers	4	7	\$5.50
Crowbar	5	2	\$7.00

- When you have gotten the programs working:
 - If you are doing this Lab synchronously, call the instructor to demonstrate its functioning.
 - If you are doing this Lab asynchronously, please submit this to Canvas.

Customer Accounts File Management System

- The structure should be used to store Inventory in a file. The program should have a menu that lets the user perform the following operations:
 1. Enter new records into the file,
 2. Display the contents of the entire file.
 3. Search for a particular item's record and change it.
 4. Search for a particular item's record and display it.
 5. Search for a particular item's record and delete it.
- *Input Validation: When the data for a new item is entered, be sure the user enters data for all the fields. No negative prices should be entered.*

Inventory File Management System

- All the modules, except the delete record module, are essentially demonstrated in Programs 12-20, 12-21 and 12-22.
- The delete record module is the only one not given to you.
- To make the delete process simpler, we will not be actually deleting a record, but rather we will be resetting it so that the first character in the description is reset to '\0', any integers are reset to 0, and the floating point numbers are reset to 0.0.
 - Note, you can assume that the ID_Number is an integer.
- This makes the delete module very similar to the Edit module, the only difference being that your program will simply put a null in for the description, '\0', and zeroes in for the numbers.

Strategy for doing Laboratory12B

- You could do this lab by simply writing the five programs indicated.
 1. Enter new blank records into the file,
 2. Display the contents of the entire file.
 3. Search for a particular item's record and change it.
 4. Search for a particular item's record and display it.
 5. Search for a particular item's record and delete it.
- However, extra credit will be given if you put them all together into one menu driven program as shown on the next page.
- Put your name at the top of the menu should you choose to do this part.

Inventory File Management System

The opening screen should display the following menu:

Main Menu

1. Create file of empty inventory
2. Display the contents of all the items
3. Make changes to a particular item
4. Search and display a particular item
5. Delete an item
6. Quit

Inventory File Management System

- Call your composite program file: *YourName*-Lab12B.cpp.
 - If you are doing this Lab synchronously, call the instructor so that you may receive credit for having accomplished this.
 - If you are doing this Lab asynchronously, submit program to Canvas.
- You will find the following programs to be useful in writing your program solution.
- Good Luck!

The following Programs Show How various operations are performed on the Inventory Random File of Structured Data

- Program12-20: Creates the file with a set number of blank records.
- Program 12-21: Displays the entire contents of the Inventory.dat file
- Program 12-22: Allows the user to edit each individual record of the file

Program 12-20: Sets up a file of blank inventory records

```
// This program sets up a file of blank inventory records.
```

```
#include<iostream>
```

```
#include<fstream>
```

```
using namespace std;
```

```
// Constants
```

```
const int DESC_SIZE = 31;
```

```
const int NUM_RECORDS = 5;
```

```
// Declaration of InventoryItem structure
```

```
struct InventoryItem
```

```
{
```

```
    char desc[DESC_SIZE];
```

```
    int qty;
```

```
    double price;
```

```
};
```

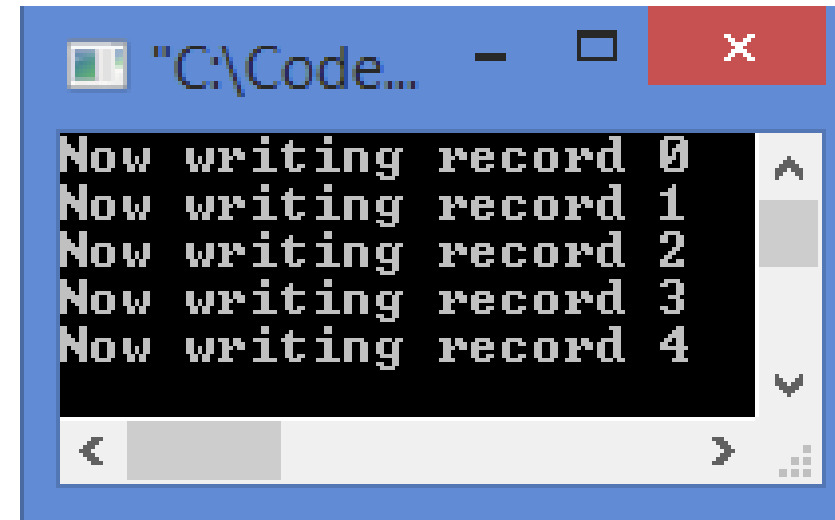
Program 12-20 Concluded

```
int main()
{
    // Create an empty InventoryItem structure.
    InventoryItem record = { "", 0, 0.0 };

    // Open the file for binary output.
    fstream inventory("Inventory.dat", ios::out | ios::binary);

    // Write the blank records.
    for (int count = 0; count < NUM_RECORDS; count++)
    {
        cout << "Now writing record " << count << endl;
        inventory.write(reinterpret_cast<char *>(&record),
                        sizeof(record));
    }

    // Close the file.
    inventory.close();
    return 0;
}
```



```
"C:\Code..."
Now writing record 0
Now writing record 1
Now writing record 2
Now writing record 3
Now writing record 4
```

Program 12-21: Displays contents of the Inventory File

```
// This program displays the contents of the inventory file.
```

```
#include<iostream>
```

```
#include<fstream>
```

```
using namespace std;
```

```
const int DESC_SIZE = 31;    // Description size
```

```
// Declaration of InventoryItem structure
```

```
struct InventoryItem
```

```
{
```

```
    char desc[DESC_SIZE];
```

```
    int qty;
```

```
    double price;
```

```
};
```

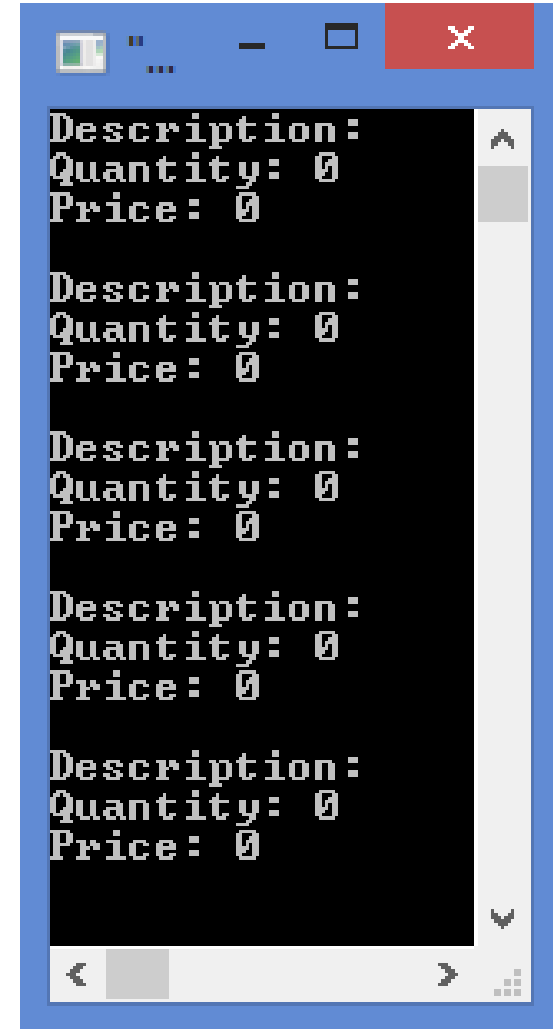
```

int main()
{
    InventoryItem record;          // To hold an inventory record
    // Open the file for binary input.
    fstream inventory("Inventory.dat", ios::in | ios::binary);
    // Now read and display the records.
    inventory.read(reinterpret_cast<char *>(&record), sizeof(record));
    while (!inventory.eof())
    {
        cout << "Description: ";
        cout << record.desc << endl;
        cout << "Quantity: ";
        cout << record.qty << endl;
        cout << "Price: ";
        cout << record.price << endl << endl;
        inventory.read(reinterpret_cast<char *>(&record),
                        sizeof(record));
    }

    // Close the file.
    inventory.close();
    return 0;
}

```

Program 12-21 Concluded



```

Description:
Quantity: 0
Price: 0

Description:
Quantity: 0
Price: 0

Description:
Quantity: 0
Price: 0

Description:
Quantity: 0
Price: 0

Description:
Quantity: 0
Price: 0

```

```
// This program allows the user to edit a specific record.
#include<iostream>
#include<fstream>
using namespace std;
const int DESC_SIZE = 31;    // Description size
// Declaration of InventoryItem structure
struct InventoryItem
{
    char desc[DESC_SIZE];
    int qty;
    double price;
};
int main()
{
    InventoryItem record;           // To hold an inventory record
    long recNum;                    // To hold a record number
    // Open the file in binary mode for input and output.
    fstream inventory("Inventory.dat", ios::in | ios::out | ios::binary);
    // Get the record number of the desired record.
    cout << "Which record do you want to edit? ";
    cin >> recNum;
    // Move to the record and read it.
    inventory.seekg(recNum * sizeof(record), ios::beg);
    inventory.read(reinterpret_cast<char *>(&record), sizeof(record));
```

Program 12-22: Program allows Editing to a Specific Record

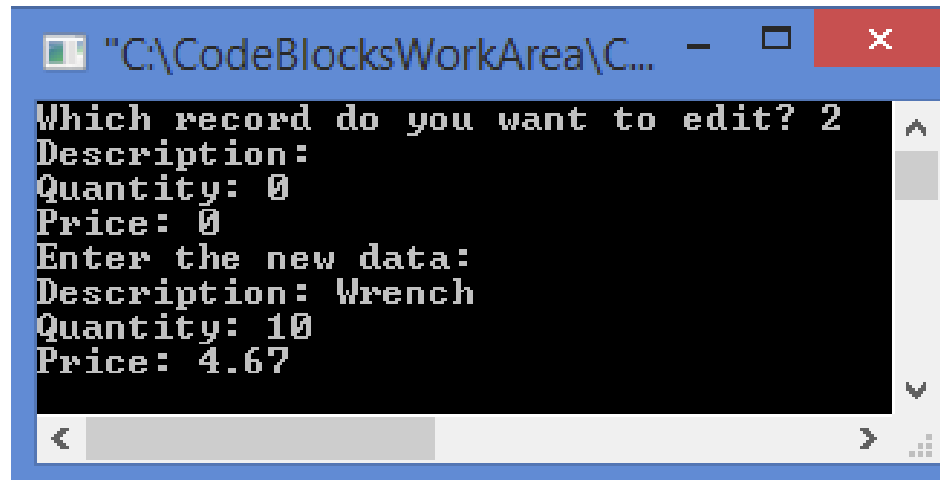
```

// Display the record contents.
cout << "Description: ";
cout << record.desc << endl;
cout << "Quantity: ";
cout << record.qty << endl;
cout << "Price: ";
cout << record.price << endl;
// Get the new record data.
cout << "Enter the new data:\n";
cout << "Description: ";
cin.ignore();
cin.getline(record.desc, DESC_SIZE);
cout << "Quantity: ";
cin >> record.qty;
cout << "Price: ";
cin >> record.price;
// Move back to the beginning of the this record's position.
inventory.seekp(recNum * sizeof(record), ios::beg);
// Write the new record over the current record.
inventory.write(reinterpret_cast<char *>(&record), sizeof(record));
// Close the file.
inventory.close();
return 0;

```

}

Program 12-22 Concluded



```

"C:\CodeBlocksWorkArea\C...
Which record do you want to edit? 2
Description:
Quantity: 0
Price: 0
Enter the new data:
Description: Wrench
Quantity: 10
Price: 4.67

```