

## Lab 1

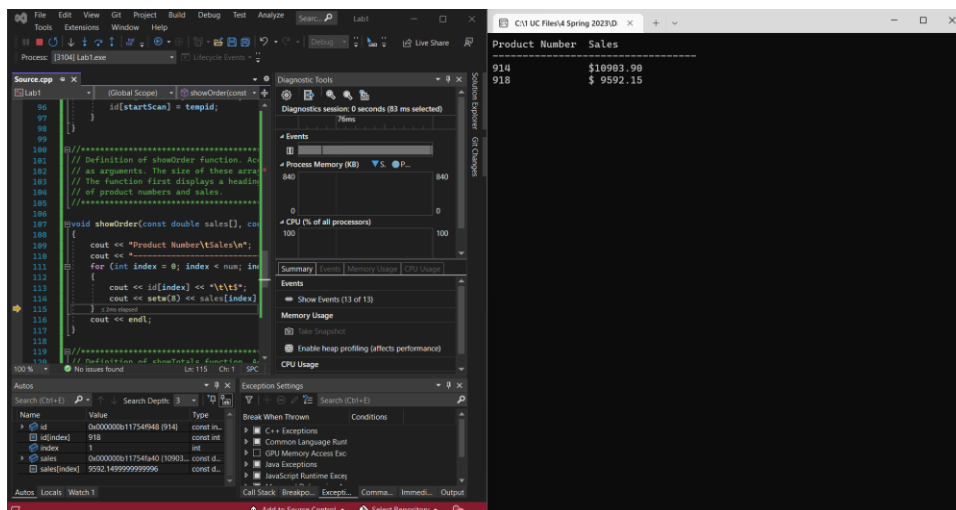
Nathan Burns and Sam Graler

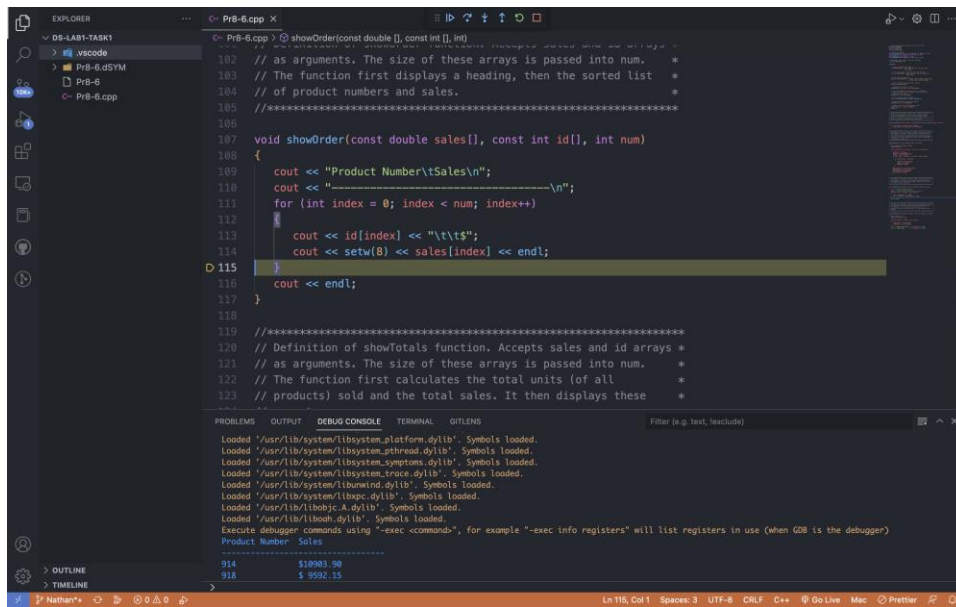
# Task 1

**A description of the objectives/concepts explored in this assignment including why you think they are important to this course and a career in CS and/or Engineering.**

When completing this assignment, we worked on different topics such as debugging and structures. We used debugging tools such as breakpoints to handle Tasks 1 and 2 by stepping through the code to find errors / bugs. This skill will translate into later labs and homework assignments when we get stuck with difficult-to-find errors. These skills will also help in future CS careers when faced with errors in the company code. It is extremely important to be able to read code that is not yours and debug it effectively, which is what we practiced in this assignment. Debugging code, you did not write also helps you learn what makes code readable to other people. Writing readable code is another critical skill in the industry. In task 3 we dealt with converting 4 different arrays into a single struct array. Knowing how to make this conversion is crucial when it comes to saving memory and organizing your code. Additionally, binding data together is another way to make your code easier to follow. Debugging and binding data together are each critical concepts to understand for a career in CS.

**Include screen shot(s) from Task 1. (We included both since our IDEs are slightly different)**





```
102 // as arguments. The size of these arrays is passed into num.
103 // The function first displays a heading, then the sorted list
104 // of product numbers and sales.
105 //*****
106
107 void showOrder(const double sales[], const int id[], int num)
108 {
109     cout << "Product Number\tSales\n";
110     cout << "-----\n";
111     for (int index = 0; index < num; index++)
112     {
113         cout << id[index] << "\t\t$";
114         cout << setw(8) << sales[index] << endl;
115     }
116     cout << endl;
117 }
118
119 //*****
120 // Definition of showTotals function. Accepts sales and id arrays
121 // as arguments. The size of these arrays is passed into num.
122 // The function first calculates the total units (of all
123 // products) sold and the total sales. It then displays these
124 // results.
125
126 void showTotals(const double sales[], const int id[], int num)
127 {
128     double totalSales = 0.0;
129     int totalUnits = 0;
130     for (int index = 0; index < num; index++)
131     {
132         totalUnits += id[index];
133         totalSales += sales[index];
134     }
135     cout << "Total Units Sold: " << totalUnits << endl;
136     cout << "Total Sales: " << totalSales << endl;
137 }
138
139 //*****
140 // Main function
141 //*****
142 int main()
143 {
144     // *****
145     // Create arrays for product numbers and sales
146     // *****
147     const int numProducts = 10;
148     int productNumbers[numProducts];
149     double sales[numProducts];
150
151     // *****
152     // Initialize arrays
153     // *****
154     for (int index = 0; index < numProducts; index++)
155     {
156         productNumbers[index] = index + 1;
157         sales[index] = 0.0;
158     }
159
160     // *****
161     // Call showOrder function
162     // *****
163     showOrder(productNumbers, sales, numProducts);
164
165     // *****
166     // Call showTotals function
167     // *****
168     showTotals(sales, productNumbers, numProducts);
169
170     return 0;
171 }
```

Product Number Sales

| Product Number | Sales    |
|----------------|----------|
| 1              | 12000.00 |
| 2              | 9500.00  |
| 3              | 15000.00 |
| 4              | 18000.00 |
| 5              | 22000.00 |
| 6              | 25000.00 |
| 7              | 28000.00 |
| 8              | 32000.00 |
| 9              | 35000.00 |
| 10             | 38000.00 |

Total Units Sold: 55  
Total Sales: 200000.00

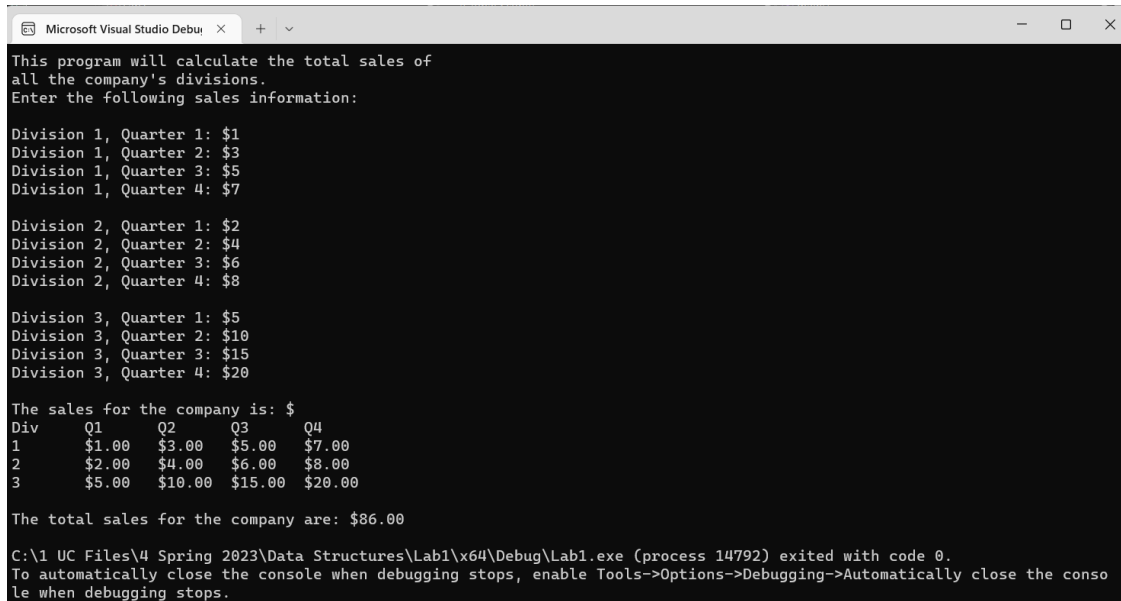
## Task 2

**A description of how you approached debugging Task 2, why you think a programmer may have made the mistakes and how you think they can be avoided in the future.**

The first thing we did to debug task 2 was run the program and examine the output. Using the description of the errors, we were able to easily locate their locations in the output. The first error we decided to find was the fact that the division in the output was incorrect. In order to fix this, we navigated through the code until we found the final for loop that outputs the data. It was here that we noticed that instead of outputting the counter "div" + 1 to display the correct division, the program was outputting the total number of divisions + 1. This mistake may have been made because the programmer was moving too quickly, and simply typed the wrong variable. The second error was hard to miss, as the large negative number was clearly garbage. This most likely indicated a memory issue, where the program tried to access something, it shouldn't have. From our experience in C++, we guessed that it was probably caused by an incorrect array index. Once again, we looked through the code until we reached the output section, and it was there we realized the bounds of the arrays had been switched. This is a very common error when working with 2D arrays that has happened to me before, and it results from forgetting which dimension you put first when declaring the array. This mistake is easily avoided by going back to check your own code and figure out what you put originally. The final error was easy to find. By simply hovering the mouse over the totalSales variable, we were able to see that it was only ever initialized and output. There were no other statements using that variable which is why it output 0. To fix this a simple summation statement was added into the

first for loop. This was mostly likely just a line that the programmer put off until later and forgot to write, and it could be avoided by slowing down. In general, these errors could be avoided by taking more time to think through the code instead of just typing it out and moving on without a second thought.

### Include screen shot(s) from Task 2.



```
This program will calculate the total sales of
all the company's divisions.
Enter the following sales information:

Division 1, Quarter 1: $1
Division 1, Quarter 2: $3
Division 1, Quarter 3: $5
Division 1, Quarter 4: $7

Division 2, Quarter 1: $2
Division 2, Quarter 2: $4
Division 2, Quarter 3: $6
Division 2, Quarter 4: $8

Division 3, Quarter 1: $5
Division 3, Quarter 2: $10
Division 3, Quarter 3: $15
Division 3, Quarter 4: $20

The sales for the company is: $
Div   Q1    Q2    Q3    Q4
1     $1.00 $3.00 $5.00 $7.00
2     $2.00 $4.00 $6.00 $8.00
3     $5.00 $10.00 $15.00 $20.00

The total sales for the company are: $86.00

C:\1 UC Files\4 Spring 2023\Data Structures\Lab1\64\Debug\Lab1.exe (process 14792) exited with code 0.
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the console when debugging stops.
```

## Task 3

### A description of what you had to do in Task 3, including any bugs you may have introduced and had to fix.

In Task 3, we had to create a structure that incorporated each of the four given variables (previously implemented using arrays) as members. That would allow us to make a single array of our product structure to replace the four arrays from the original. We then had to initialize the structure using the values from each of the arrays. Finally, we had to modify the functions and their prototypes to accept the structure as input, rather than the individual arrays (the size argument was kept in each function). However, this change resulted in errors within the method's code, so we needed to replace all references to the arrays with the corresponding variables within the structure. The major bugs we faced were errors when we forgot to change all three lines that describe a function and its parameters / arguments (the prototype, the call, and the header when writing the actual function). There was also a logic error at first when re-implementing the Dual Sort function. Originally, the code only switched the sales and IDs to reflect the proper order, but when using a structure, you must keep all relevant data bound together. This was accomplished by creating a temporary product (rather than a temporary id and sale variable) and switching entire products to reflect the order of sales.

### Include screen shot(s) from Task 3.

```
Microsoft Visual Studio Debug Console
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

C:\1 UC Files\4 Spring 2023\Data Structures\Lab1\x64\Debug\Lab1.exe (process 16716) exited with code 0.
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the console when debugging stops.
Press any key to close this window . . .
```