Caleb Cargill

Brandon Jones

Manvith Krishna

Keith Springs

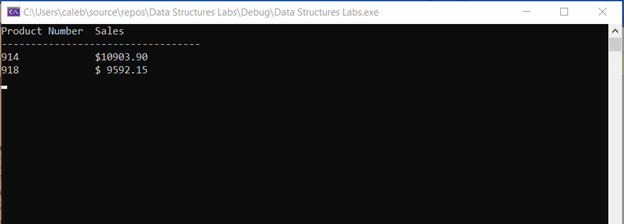
**Data Structures (2028C) – Lab 1: IDE, Debugging, Arrays, and Structs**

1. The main task of this lab was to become familiarized with useful tools that we can use with an IDE in order to solve problems with our code. Mainly, we used breakpoints in order to step through our code to see where issues were occurring within the code. Debugging is very useful for programmers because it is almost inevitable that we will make mistakes in our code and the chances of this only increase as the length of our code increases. It is important to know how to find where the issues are occurring in the code in order to increase your efficiency as a programmer. We also were tasked with creating a struct in Task 3 which is another extremely useful tool for programmers. Structs allow you to group together similar data which keeps the code concise and easy to follow which is especially important when working with a group of people which is extremely common in industry.
2. The first thing we did was run the program as it first was to see what was going wrong. The program ran without any compilation errors, but it was not outputting the correct data as described in the task in Task 2. First thing we did was add a breakpoint at the first nested for loop that added the values to the 2d array. We did not see any issues with this code. Next, we added a breakpoint at the second nested for loop that displayed the quarterly sales for each division. In this nested loop, we realized that in the cout statement, it was using sales[qtr][div] which is backwards from how it was supposed to be. Because of this, it was drawing in garbage data since the indexes were wrong and once it was changed to sales[div][qtr], this issue was fixed. The total sales and the division values were still wrong after this fix. We noticed that totalSales is defined and set to the value 0 but is never actually used in the code. To fix this, we added totalSales = totalSales + sales[div][qtr] within the nested for loop that displays the quarterly sales. This statement could have also gone in the first nested loop that added the values to that array. This fixed the issue with the total sales being 0. The last issue was the division number always being 4 when outputting the information. To fix this we looked in the output loop and noticed that there was a statement cout << NUM\_DIVS + 1 << "\t" and NUM\_DIVS is 3 so that is why the division was always 4. To fix this, you just replace NUM\_DIVS with divs so that it iterates through each number and get the statement cout << div + 1 << "\t". This could have also been found using a breakpoint if we didn’t notice it initially. Now all issues were fixed, and the code was running how it was supposed to.
3. For Task 3, we simply made a struct called products that has 4 member variables to represent the 4 arrays that it is replacing. We also made it so that instead of the functions taking a value from each of the 4 arrays, it now just takes a products struct as a parameter. After doing this, we assigned each element of the vector of our products struct with the corresponding unit and price values from the initial units and prices arrays. Basically, anywhere that used a value from one of the 4 original arrays, we replaced it with using the corresponding variable from our products struct vector instead.

**Screenshots**

**Task 1:**

Output:

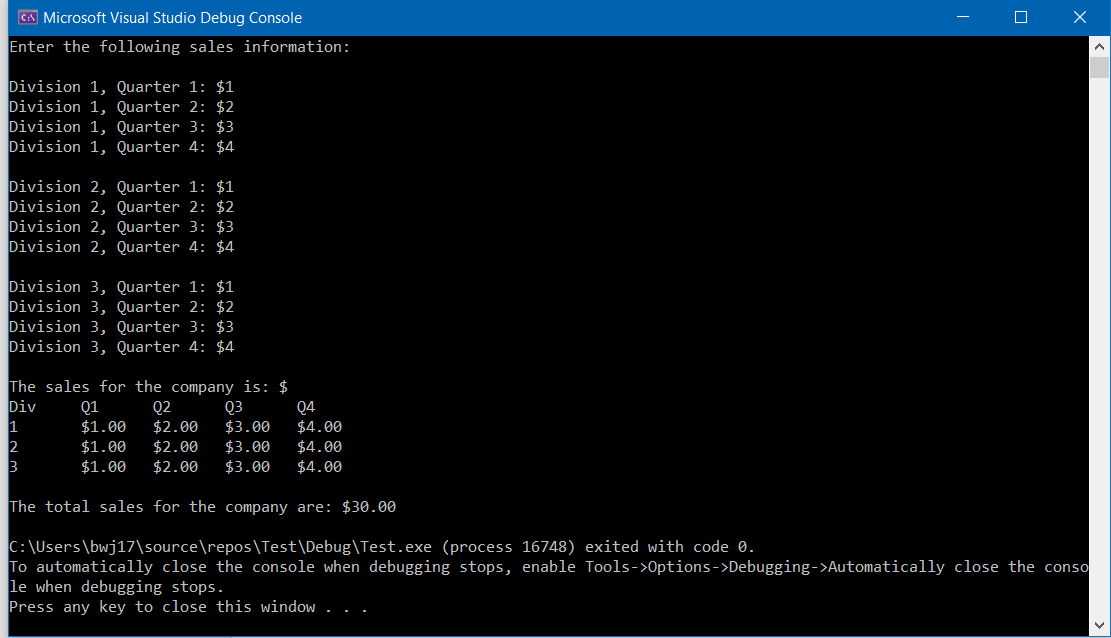


A screenshot of a computer screen

Description automatically generated IDE:

**Task 2:**

Output:



**Task 3:**

Output:

