Katerina Betts and Donald Huchinson

Professor Charles Zimmer

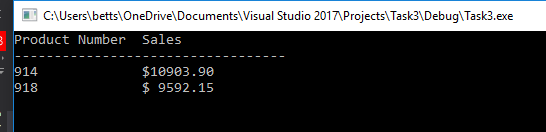
Data Structures

23 January 2018

Lab One Report

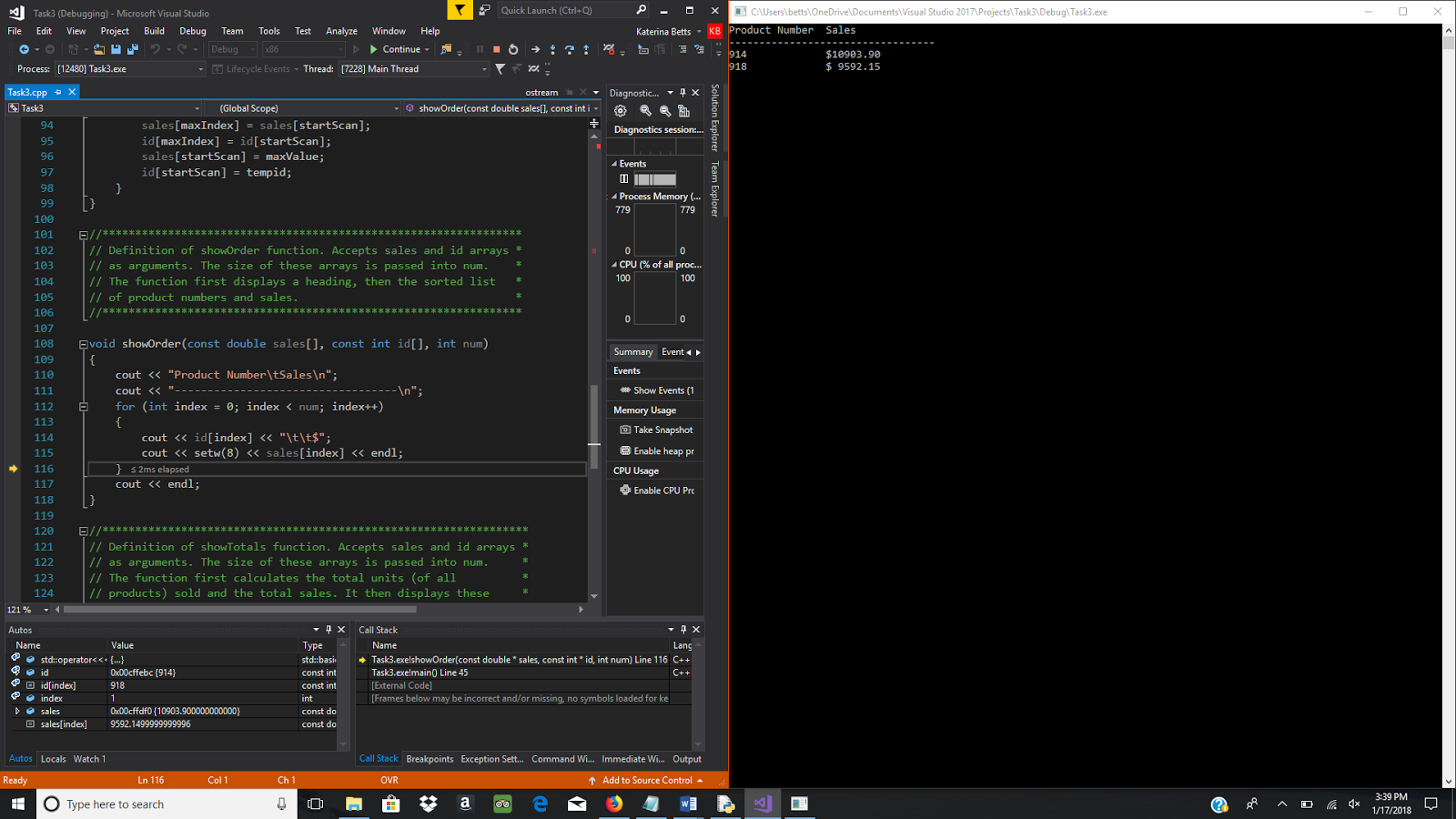
In the first lab in Data Structures, the objective was to fix code that was improperly written while using the debugger on a specific IDE to do so. It was also important to figure out how to change the multiples arrays in the code into one easily useable array by utilizing a structure. The debugging process utilized in this lab is important to a career in computer science as a software engineer or developer. The ability overall to be able to successfully use a debugger is not only important when working with code that is written by a single person, but it especially important when working with the code of a team. The accurate usage of the debugging system helps the programmer and his or her team to quickly assess the situation as opposed to search through hundreds of lines of code to potentially find one small syntax error that is throwing the entire program into compile or runtime errors. This concept mostly applies to Task One and Task Two in this lab, but it could also be applied to Task Three where the already written code needs to be put into a simpler format. When it comes to putting the multiple arrays into a structure in order to group like items and take up less space, this is an important concept to learn. It not only makes the code easier to read, it also puts all four of the arrays that were originally in the code in one place that does not change and is easier to find for those who have not worked with the code in Task 3.

Figure 1: Output in Task One



*This is a view of the values output when following the instructions provided in Task One.*

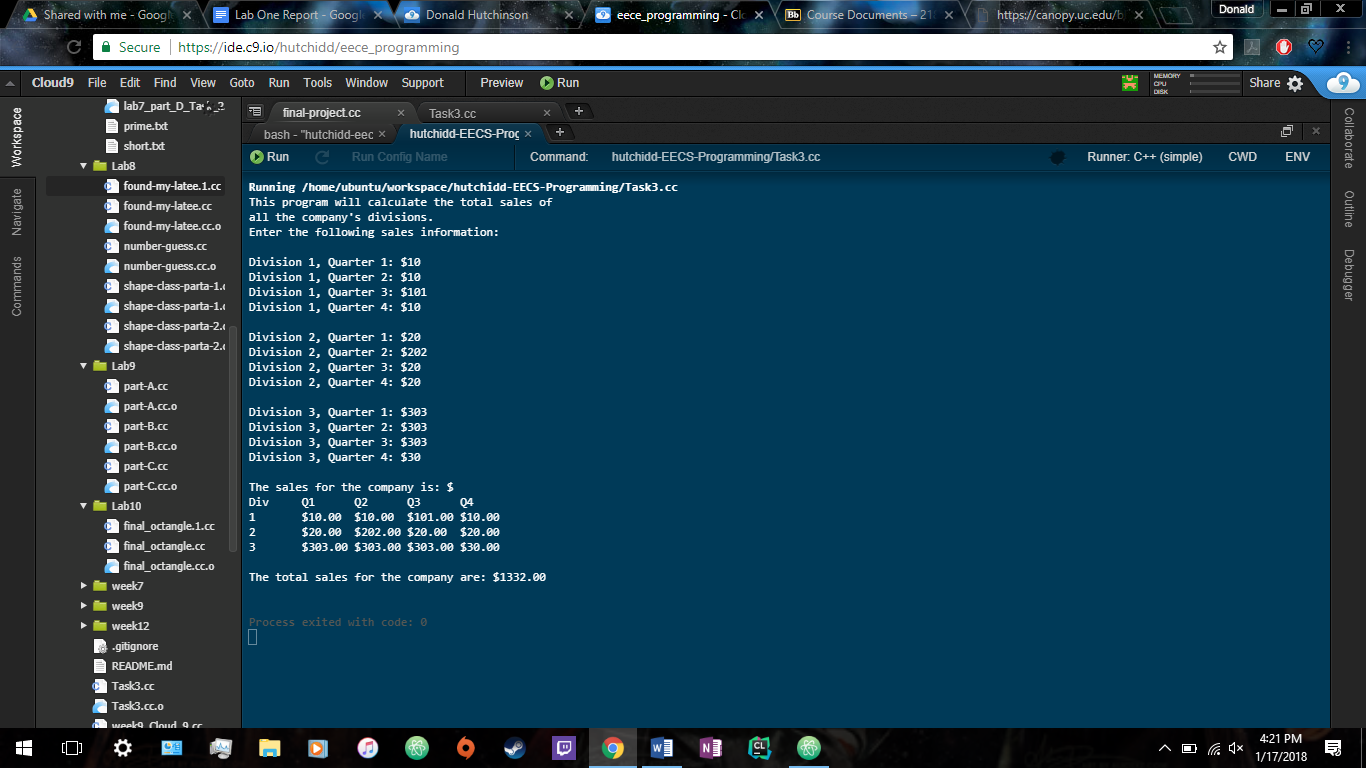
Figure 2: Further Output in Task One



*The IDE used is on the left side of the screen while the output of the code is shown on the right.*

In Task Two, the program needed to be debugged as it was outputting incorrect data and large negative numbers that were impossible given the scenario. Using the debugging methods described in class along with prior knowledge of debugging, Task Two was fixed by first taking a look at the code as a whole and then going into it piece by piece with the specific IDE’s debugging system. For the mistake that was initially on line 39 of the code, the mistake was made because div and NUM\_DIVS are very similar variable names for one set of code. This could have been avoided had the original programmer realized that NUM\_DIVS not only followed what is the typical way to write a constant variable, but it was, in fact, a constant variable. A quick check of the variables could have fixed this problem. The second error, the one on line 42, happened due to a switch in the sales matrix. This is a mistake that is easily made when trying to type code as quickly as possible. Of course, the error would easily be avoided if the programmer would have checked to make sure the written instance of sales matched with the original instance, which was sales[div][qtr] instead of sales[qtr][div]. The last error occurs within a loop due to the nonexistent update to totalSales when a user enters a value. This can be easily remedied by adding another line to the code in order to fix the nested for loop, but this is another mistake that could have been avoided by testing the code before finishing it or potentially debugging it earlier on in the coding process.

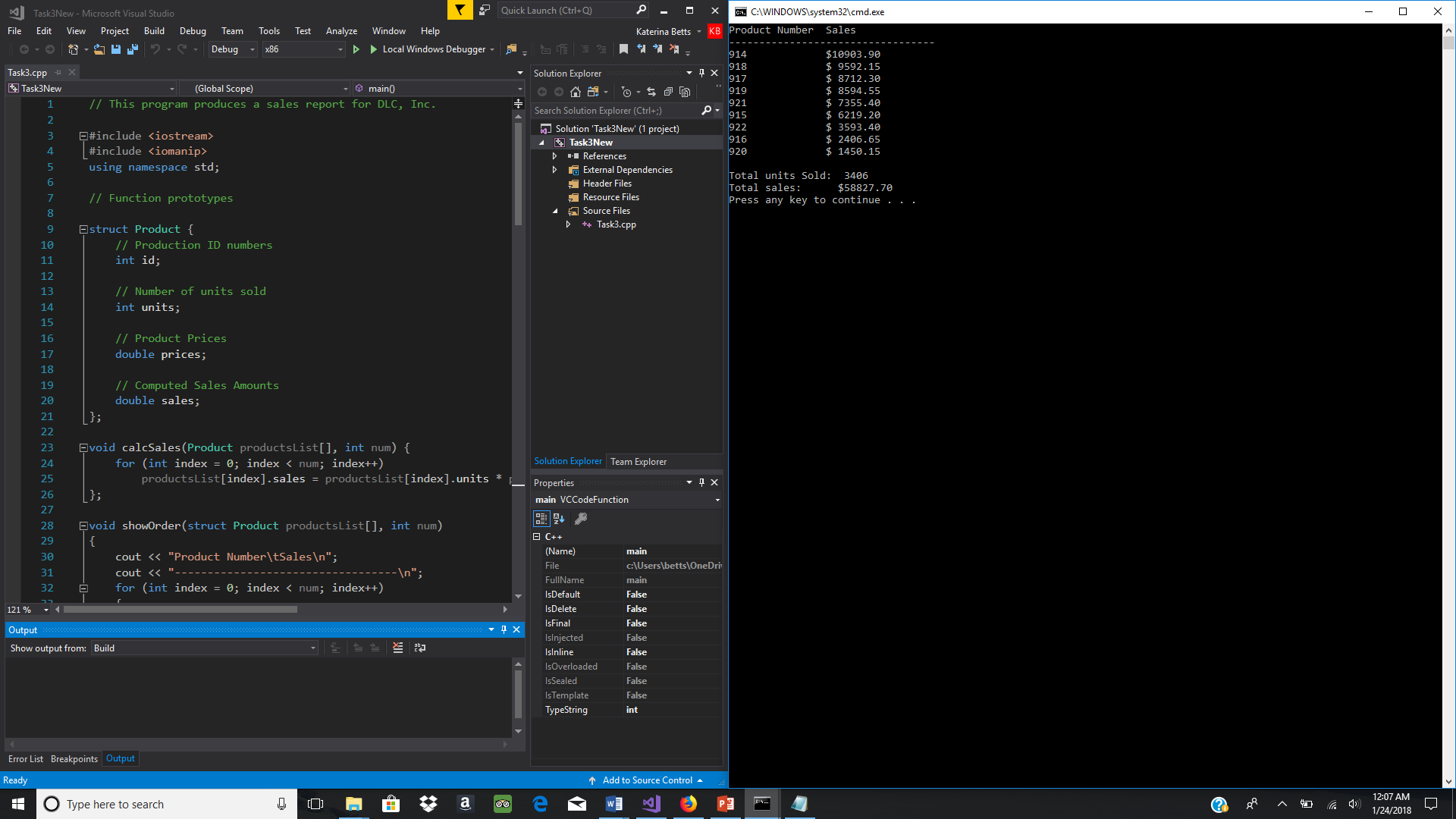
Figure 3: Output in Task Two



*The fixed output in Task Two is shown on the screen in a different IDE than used earlier.*

In Task Three, the goal was to move the four arrays that were present in the code into a structure to contain them while also modifying the existing functions to work with the new structure. This created a few new problems as the syntax of adding structures into functions was not as well known and multiple parts of the functions had to be changed to allow for the usage of a structure instead. The biggest error and issue in this part of the lab was the missing of a variable change. If just one variable was not in the form of the structure, the entire program would refuse to compile and would through a few errors. However, other than that part of the process, the switch from four arrays to one structure of arrays was completed with minimal errors and no bugs.

Figure 4: Output of Task Three



*Shown above is the output in Task Three with the four arrays put into a structure.*