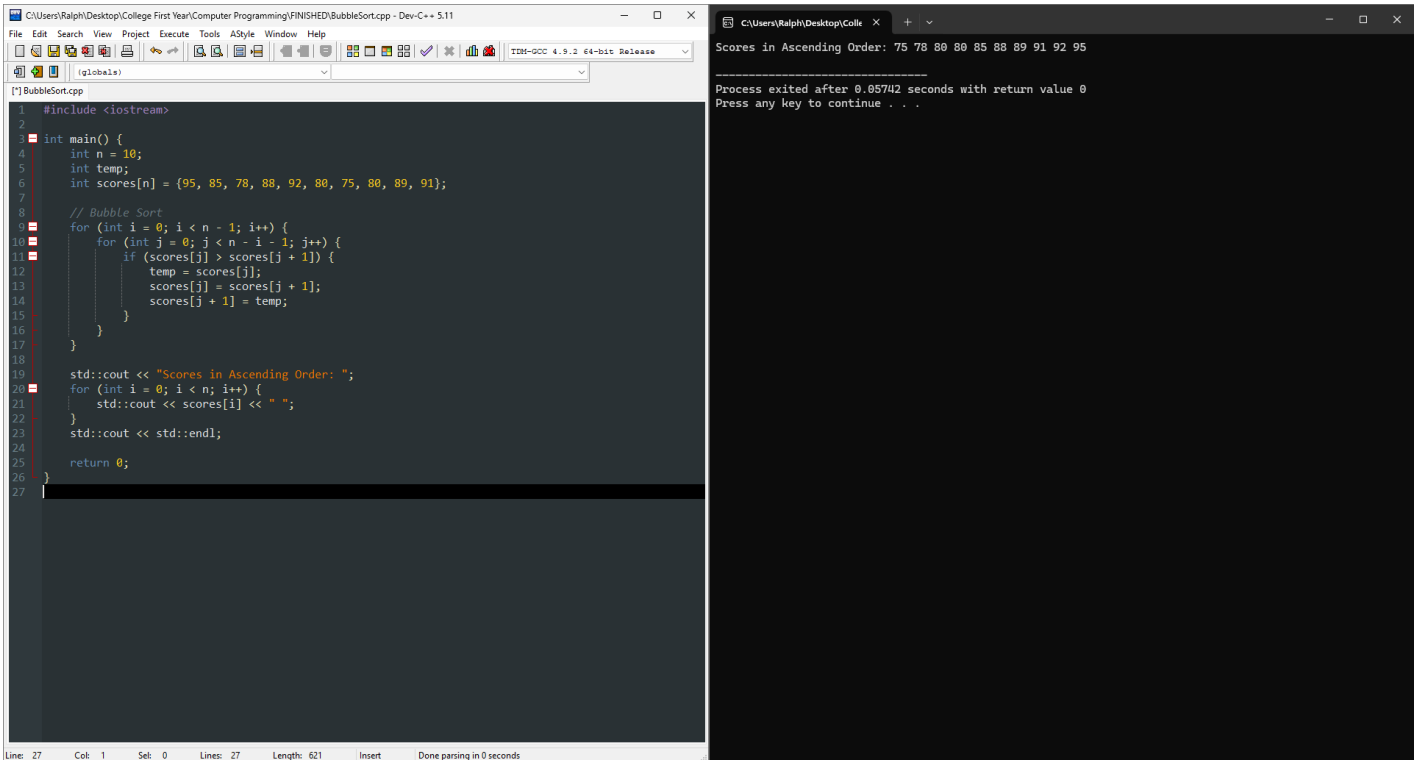
Assignment 4.2: Bubble Sort	
Bubble Sort	
Course Code: CPE007	**Program:** Computer Engineering
Course Title: Programming Logic & Design	**Date Performed: 11/9/2025**
Section: CPE11S1	**Date Submitted: 11/9/2025**
Name(s) Ralph Angelov F. Braganza	**Instructor: Engr. Jimlord M. Quejado**
Output	

BUBBLE SORT CODE:

```
#include <iostream>

int main() {
    int n = 10;
    int temp;
    int scores[n] = {95, 85, 78, 88, 92, 80, 75, 80, 89, 91};

    // Bubble Sort
    for (int i = 0; i < n - 1; i++) {
        for (int j = 0; j < n - i - 1; j++) {
            if (scores[j] > scores[j + 1]) {
                temp = scores[j];
                scores[j] = scores[j + 1];
                scores[j + 1] = temp;
            }
        }
    }

    std::cout << "Scores in Ascending Order: ";
    for (int i = 0; i < n; i++) {
        std::cout << scores[i] << " ";
    }
    std::cout << std::endl;

    return 0;
}
```

RESULT:



Discussion:

Bubble Sort is the simplest sorting algorithm that works by repeatedly swapping the adjacent elements if they are in the wrong order (GeeksforGeeks, 2025).

For a clear understanding of the code we must breakdown the code into pieces/step by step. We first have to initialize our variables, scores [] will hold our 10 integers (95, 85, 78, 88, 92, 80, 75, 80, 89, 91) and between [ ] we will put 10 or we could simply assign n = 10 so that it will be easier for us to identify what “n” is for when reading the code. Let us now talk about our loops, the outer for-loop runs n-1 this will be able to “push” the largest remaining element at the end. The inner for-loop, this loop compares the adjacent elements, so basically scores[j] is the current element while the one next to that is scores[j+1] (we added 1 so it goes to the next element) and scores[j] > scores[j+1] will check if the current is bigger than the next one and if that proves to be true its out of order because we are trying to make it into ascending order (smaller comes first). This is where our initialized temp comes in, it stores scores [j] so it will be able to move the smaller element which is scores[j+1] into its current spot (scores[j]). After it does that whole process it will first print out our text "Scores in Ascending Order: " and start printing out the scores. It will start with the first index i = 0 of the array then it will keep going until i < n (it reaches the number of elements we have in our array) and i++ just moves to the next index we have in our array. We added “ “ at the end of our std::cout so the numbers will have a space between. That is the end of the code and how it generally works.

Supplementary Activity

Conclusion

As always, every activity is a challenge, and me trying to explain this in depth was difficult, but at the end I managed to understand the concepts of how the code operates even more now after analyzing it. It took me a while to be able to understand how the temp can swap the scores[j], and it took me a whole lot of time trying to understand each loop and how they work to be able to sort the array in ascending order. Overall, this activity has given me a guide of how bubble sorts work, and even though bubble sort is the simplest sorting algorithm, I’m quite scared of what other algorithms there are out there in the coding world. It may be difficult, but I have to face the fact that I chose computer engineering, and this

is something I must endure to be able to improve. The activity will serve as my foundation as I progress to more challenging tasks in the future.