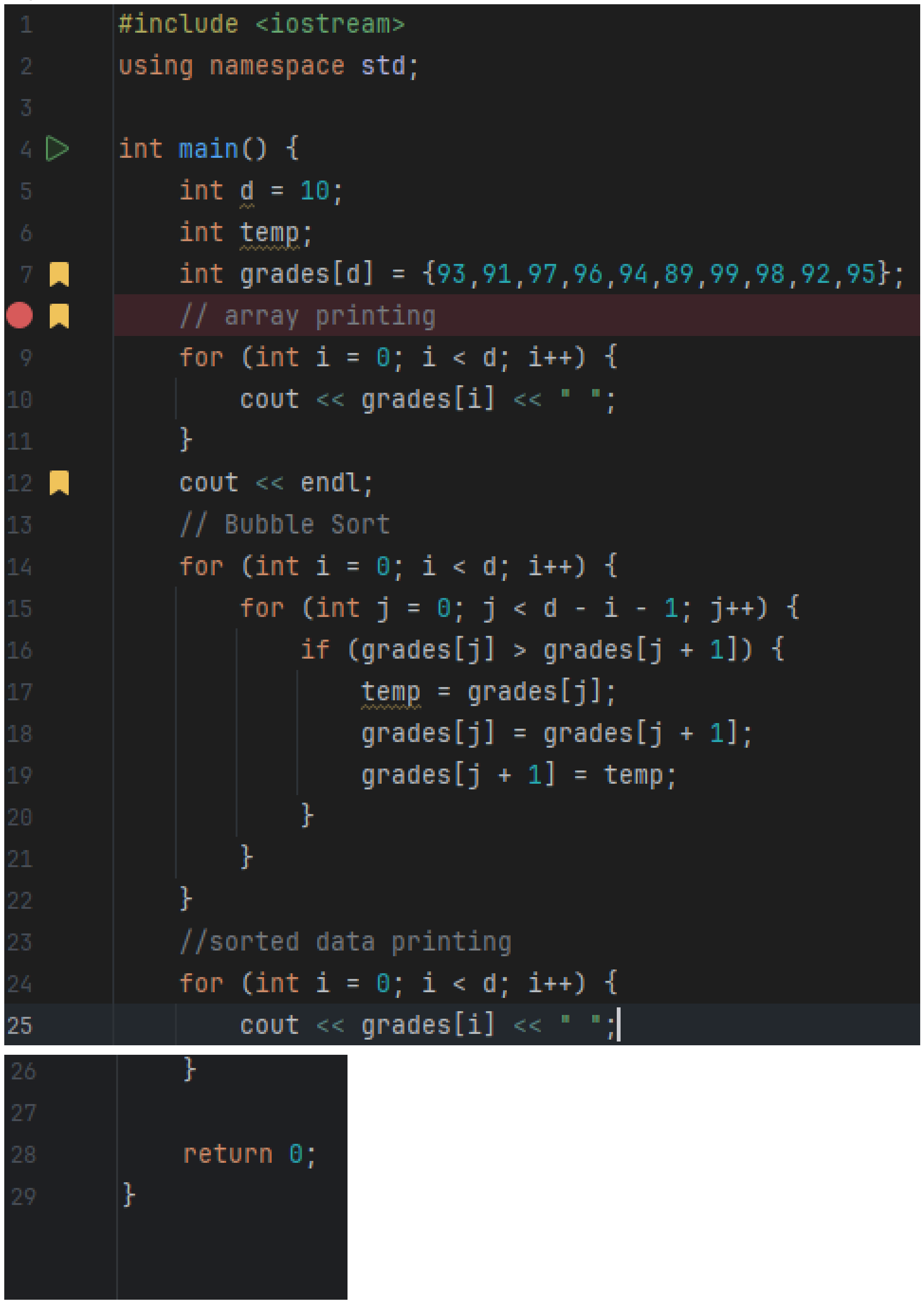
Assignment 4.2	
Bubble Sort	
Course Code: CPE007	**Program:** Computer Engineering
Course Title: Programming Logic and Design	**Date Performed: September 11, 2025**
Section: CPE11S1	**Date Submitted: September 11, 2025**
Name(s): Jaime Luis M. Demain	**Instructor: Engr. Jimlord M. Quejado**
6. Output	

Implementation Code:

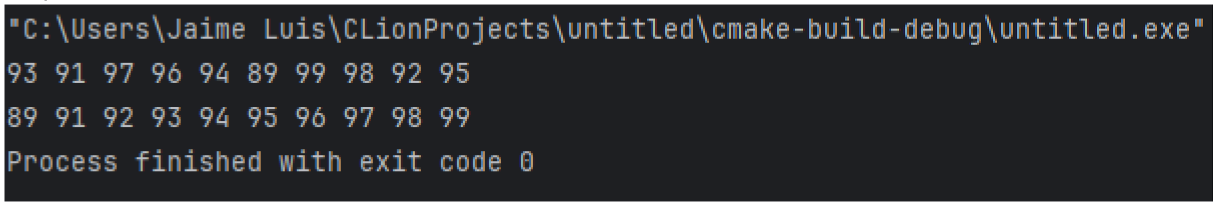


```cpp
#include <iostream>
using namespace std;

int main() {
    int d = 10;
    int temp;
    int grades[d] = {93,91,97,96,94,89,99,98,92,95};
    // array printing
    for (int i = 0; i < d; i++) {
        cout << grades[i] << " ";
    }
    cout << endl;
    // Bubble Sort
    for (int i = 0; i < d; i++) {
        for (int j = 0; j < d - i - 1; j++) {
            if (grades[j] > grades[j + 1]) {
                temp = grades[j];
                grades[j] = grades[j + 1];
                grades[j + 1] = temp;
            }
        }
    }
    //sorted data printing
    for (int i = 0; i < d; i++) {
        cout << grades[i] << " ";
    }

    return 0;
}
```

Output:



```
"C:\Users\Jaime Luis\CLionProjects\untitled\cmake-build-debug\untitled.exe"
93 91 97 96 94 89 99 98 92 95
89 91 92 93 94 95 96 97 98 99
Process finished with exit code 0
```

7. Supplementary Activity

Comprehensive Discussion:
Bubble Sort happens when you want to sort the values of index in either ascending order or descending order. It is an algorithm that sorts an array from the lowest value to the highest value and vice versa. How does it work? Well, it compares adjacent values from left to right in an array and it swaps them if they are in the wrong order. For example, we have an array of [6,8,3,5,7]. If we want to sort it out in ascending order, we will start with the first value [6] and we will compare it with the second value [8]. If we find out that these values are out of order, we will swap them using the temp language. And this process will continue until it reaches the end of the array. The largest value then will be shifted to the end of the array and the smallest value will be shifted to the beginning of an array. However, not all the cases then that the arrays will be fully sorted out in one pass. If an array is not fully sorted yet in ascending or descending order, the process will be looped until it is fully sorted out. In C++ language, there are three parts of loops to run pass, compare adjacent values, and provide a mechanism to swap values. “for (int i = 0; i < d; i + +)” provides a mechanism to control pass by ensuring that the largest element to be compared will be shifted to its correct position at the later part of the array. “for (int j = 0; j < d - i - 1; j + +)” meanwhile provides a mechanism to compare adjacent values (j and j+1) and setting a limit on until elements can be compared (d - i - 1). Lastly, “if (grades[j] > grades[j + 1])” provides a mechanism for elements to be swapped wherein it utilizes the if statement so that it can identify if the inputted condition is met and swapping elements can be executed.

Source:

W3Schools.com. (n.d.). https://www.w3schools.com/dsa/dsa_algo_bubblesort.php

Alake, R. (2025, August 12). Bubble sort time complexity and algorithm explained. Built In.

https://builtin.com/data-science/bubble-sort-time-complexity

8. Conclusion

What I learn about this activity is how bubble sort works. Bubble Sort in name refers to the way in which larger elements “bubble” to the top or the end of the array, as they are repeatedly compared and swapped with smaller elements when sorted in ascending order and vice versa. It works by comparing adjacent values and the swap will be performed if the former value is found out to be higher than the later value and the comparison will be continued until the former value reaches its correct end position of an array. If the array is not sorted out in one pass, the process will be repeated until the array is fully sorted.