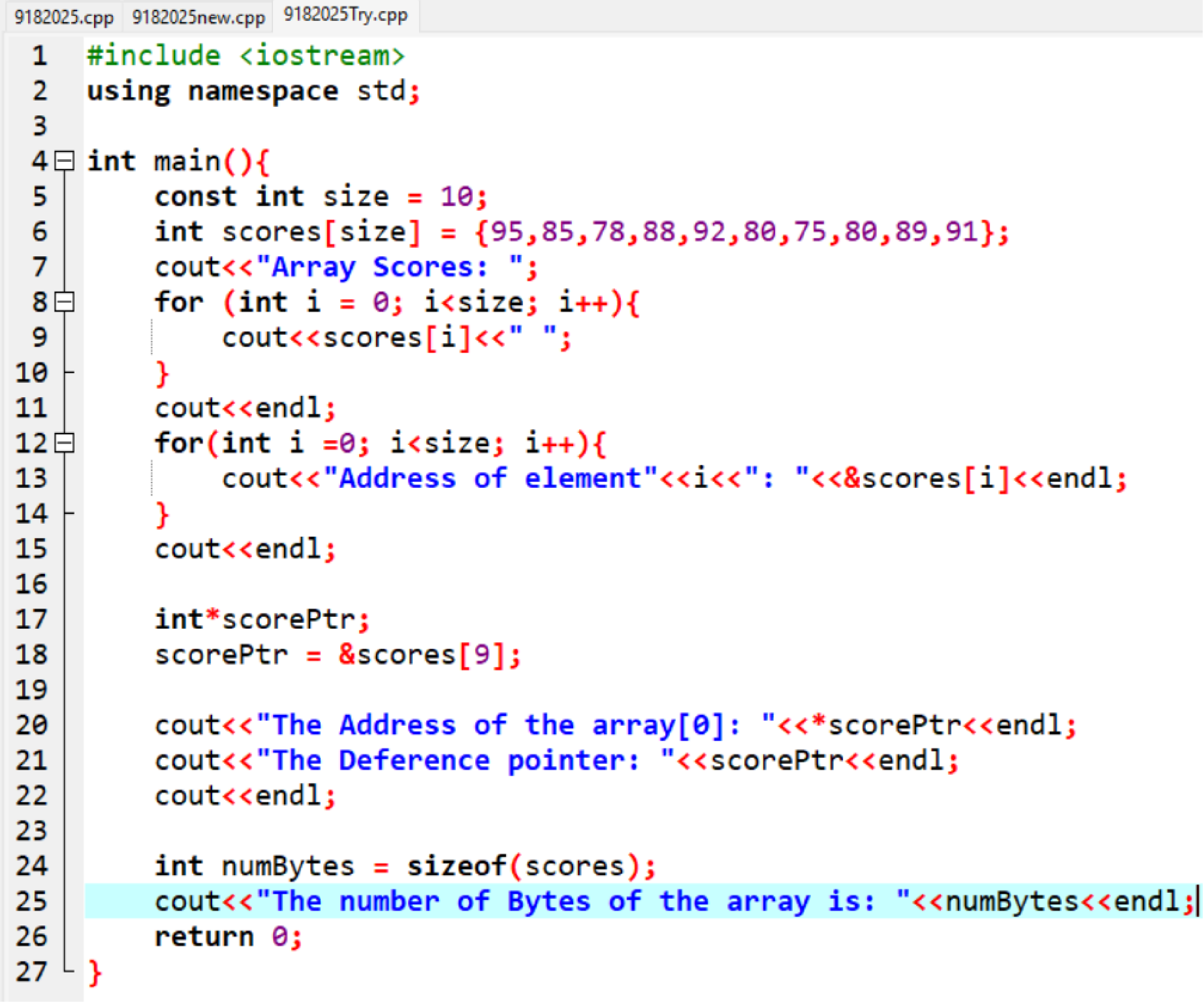
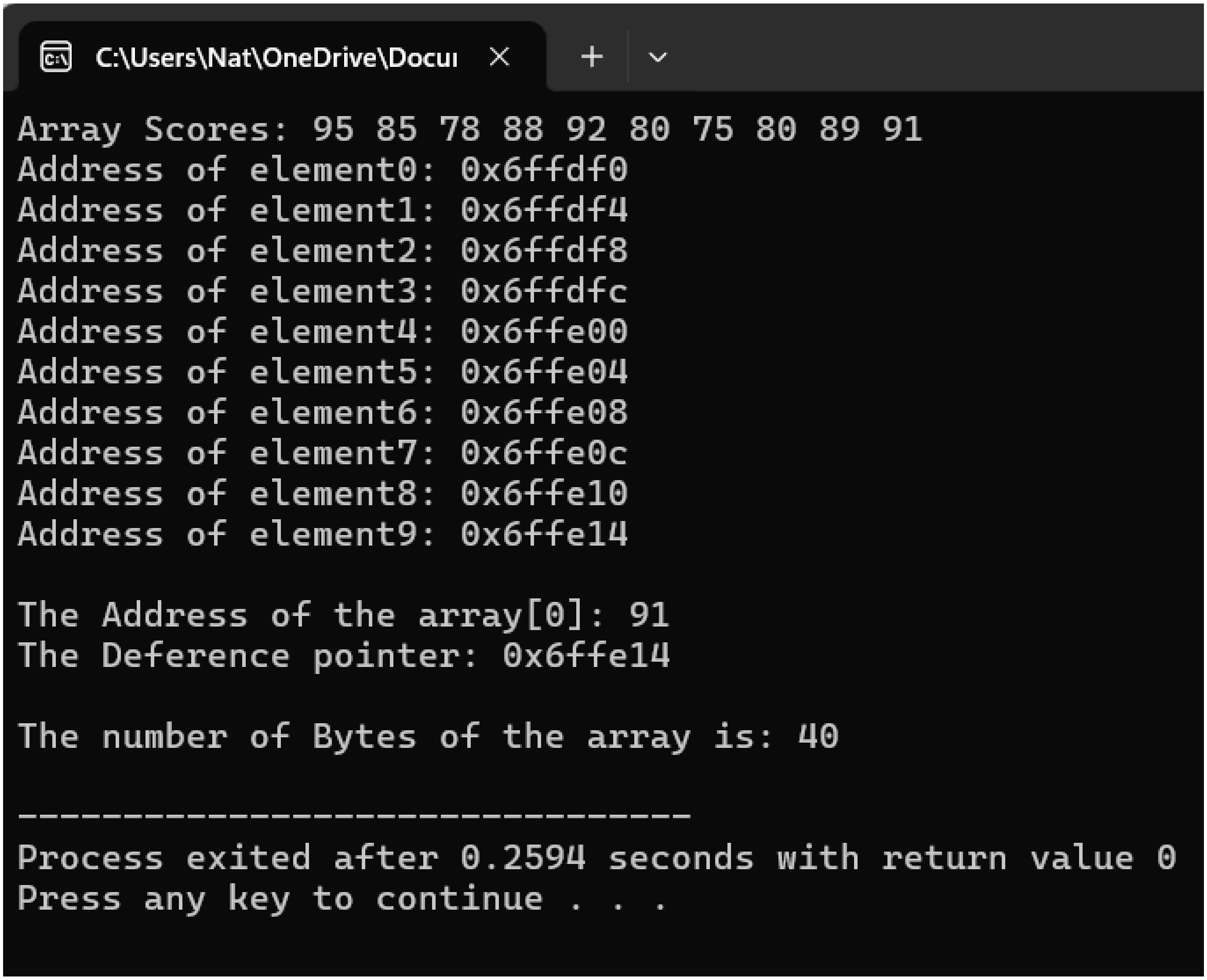
SEATWORK No. 4.2	
Pointers	
Course Code: CPE007	**Program:** Computer Engineering
Course Title: Programming Logic and Design	**Date Performed:** September 18, 2025
Section: CPE11S1	**Date Submitted:** September 18, 2025
Name(s): Mendoza, Nathaniel B.	**Instructor: Engr. Jimlord M. Quejado**
6. Output	

Code:



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

int main(){
    const int size = 10;
    int scores[size] = {95,85,78,88,92,80,75,80,89,91};
    cout<<"Array Scores: ";
    for (int i = 0; i<size; i++){
        cout<<scores[i]<<" ";
    }
    cout<<endl;
    for(int i =0; i<size; i++){
        cout<<"Address of element"<<i<<": "<<&scores[i]<<endl;
    }
    cout<<endl;

    int*scorePtr;
    scorePtr = &scores[9];

    cout<<"The Address of the array[0]: "<<*scorePtr<<endl;
    cout<<"The Deference pointer: "<<scorePtr<<endl;
    cout<<endl;

    int numBytes = sizeof(scores);
    cout<<"The number of Bytes of the array is: "<<numBytes<<endl;
    return 0;
}
```

Code Output:



```
Array Scores: 95 85 78 88 92 80 75 80 89 91
Address of element0: 0x6ffdf0
Address of element1: 0x6ffdf4
Address of element2: 0x6ffdf8
Address of element3: 0x6ffdfc
Address of element4: 0x6ffe00
Address of element5: 0x6ffe04
Address of element6: 0x6ffe08
Address of element7: 0x6ffe0c
Address of element8: 0x6ffe10
Address of element9: 0x6ffe14

The Address of the array[0]: 91
The Deference pointer: 0x6ffe14

The number of Bytes of the array is: 40

--------------------------------
Process exited after 0.2594 seconds with return value 0
Press any key to continue . . .
```

7. Supplementary Activity

Code Analysis:

The code started using #include <iostream> which is known as the Input/Output stream, allowing it to use cout for displaying outputs and cin if ever wants it to receive inputs. Next, it is implemented using namespace std; so that I won't need to repeatedly type std::cout or std::cin every time it wants to be used. Then, it wrote int main() which is the starting point of the program. Everything inside the curly braces {} will be executed when the program runs. Inside the main function, It was first declared a constant integer size = 10; which specifies the number of elements the array will hold. After that, it created an integer array scores[size] and initialized it with 10 values {95, 85, 78, 88, 92, 80, 75, 80, 89, 91}. This array represents a collection of scores that are stored in consecutive memory locations. Next, the code displayed the heading *"Array Scores: "* using cout. To print the contents of the array, then a for loop starting from *i = 0* up to *i < size*. Inside the loop, then it used cout << scores[i] << " "; which prints each element of the array followed by a space. This results in all scores being displayed in one line separated by spaces. After printing the values, it used another for loop to print the memory addresses of each array element. The loop goes through each index from *0 to 9*, and inside the loop, then it writes cout << "Address of element " << i << ": " << &scores[i] << endl;. This prints the index number and the corresponding memory address where that element is physically stored in memory. Then, the code declared a pointer variable int *scorePtr;. A pointer is a special variable that is used to store memory addresses. Then I assigned scorePtr = &scores[9]; which means that the pointer now stores the address of the last element in the array (scores[9]). After this, I used cout to display two pieces of information. First, "The Address of the array[0]:" << *scorePtr actually prints the value stored in the last element of the array because of the dereferencing operator *. This means instead of showing the address, it displays the value 91. Second, "The Dereference pointer:" << scorePtr prints the actual memory address stored in the pointer, which is the location of the 10th element (scores[9]). Next, I declared an integer variable numBytes and assigned it the value of sizeof(scores);. This calculates the total size of the array in bytes. Since the array contains 10 integers and each integer usually occupies 4 bytes in memory, the total comes out to 40 bytes using the formula so basically it is just 10bytes*(multiplied) by 4 bytes is just equal to 40. Then printed this result using cout << "The number of Bytes of the array is: " << numBytes;. Finally, It ended the program with return 0; which indicates that the program has run successfully without errors.

8. Conclusion

Today, I learned how arrays, memory addresses, and pointers are connected in C++. Meaning, I understand that I could use the following 3 in 1 code itself just by displaying both the values of the array and their respective addresses, I also realized that arrays are not only collections of numbers but also sets of memory locations stored consecutively. The use of a pointer to access the last element showed me how pointers can reference and manipulate memory directly, while dereferencing allowed me to retrieve the value stored in that location. I also discovered that using sizeof reveals how much memory the entire array takes, which depends on both the number of elements and the size of each data type.
Additionally, I learned about taking the code bytes with the use of a formula and a multiplication in which I found that bytes depend on what element I use for my code. Overall, this lesson helped me understand and made me more aware of how data is stored and accessed in C++.