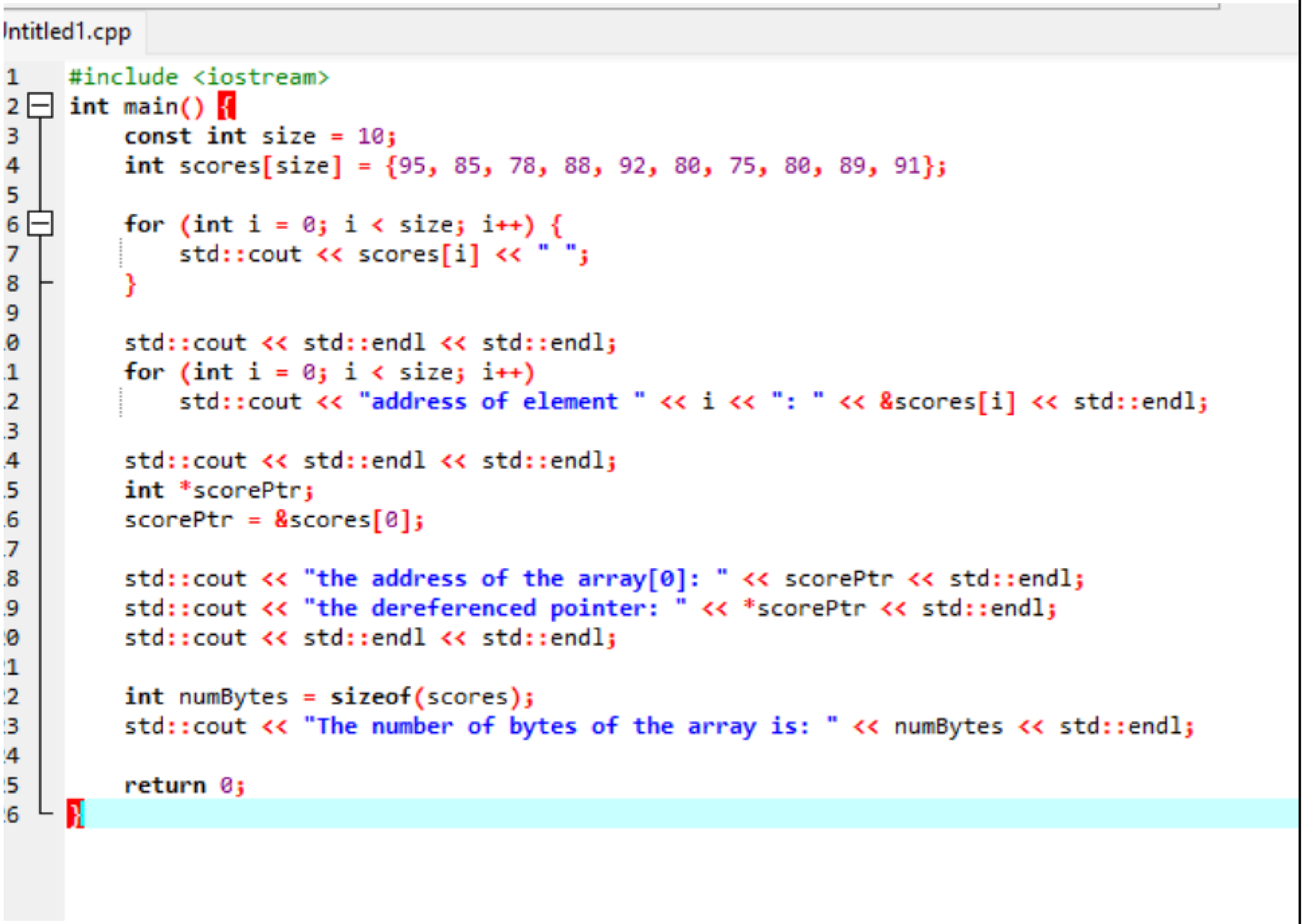
Seatwork 4.2:	
Pointers	
Course Code: CPE007	**Program:** Computer Engineering
Course Title: Programming Logic and Design	**Date Performed:** 9/18/2025
Section: CPE11S1	**Date Submitted:** 9/18/2025
Name(s): Ramirez, Angel Mae C.	**Instructor:** Engr. Jimlord M. Quejado
6. Output	

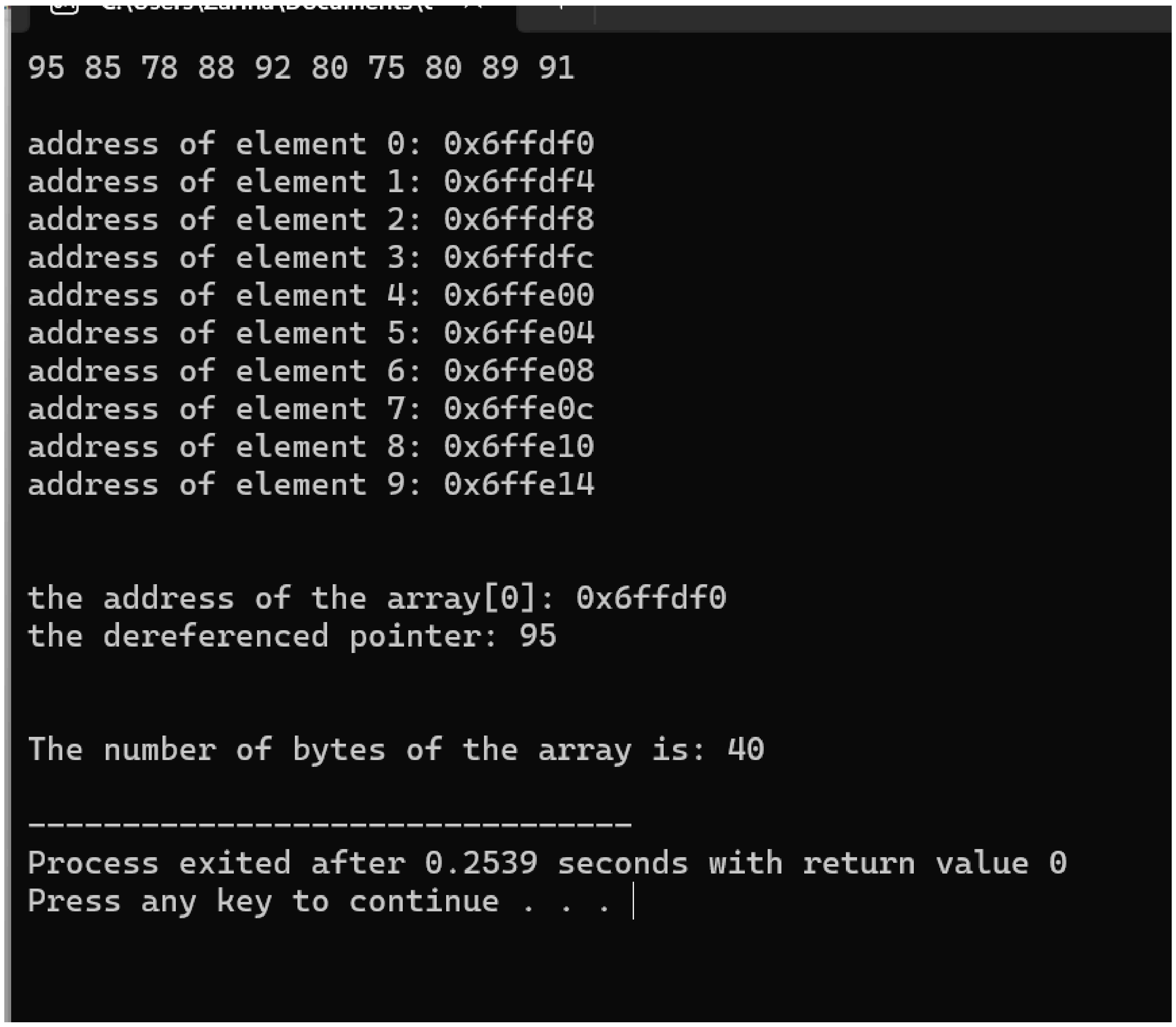
CODE:



ntitled1.cpp

```cpp
#include <iostream>
int main() {
    const int size = 10;
    int scores[size] = {95, 85, 78, 88, 92, 80, 75, 80, 89, 91};

    for (int i = 0; i < size; i++) {
        std::cout << scores[i] << " ";
    }

    std::cout << std::endl << std::endl;
    for (int i = 0; i < size; i++)
        std::cout << "address of element " << i << ": " << &scores[i] << std::endl;

    std::cout << std::endl << std::endl;
    int *scorePtr;
    scorePtr = &scores[0];

    std::cout << "the address of the array[0]: " << scorePtr << std::endl;
    std::cout << "the dereferenced pointer: " << *scorePtr << std::endl;
    std::cout << std::endl << std::endl;

    int numBytes = sizeof(scores);
    std::cout << "The number of bytes of the array is: " << numBytes << std::endl;

    return 0;
}
```

OUTPUT:



```
95 85 78 88 92 80 75 80 89 91

address of element 0: 0x6ffdf0
address of element 1: 0x6ffdf4
address of element 2: 0x6ffdf8
address of element 3: 0x6ffdfc
address of element 4: 0x6ffe00
address of element 5: 0x6ffe04
address of element 6: 0x6ffe08
address of element 7: 0x6ffe0c
address of element 8: 0x6ffe10
address of element 9: 0x6ffe14


the address of the array[0]: 0x6ffdf0
the dereferenced pointer: 95


The number of bytes of the array is: 40

--------------------------------
Process exited after 0.2539 seconds with return value 0
Press any key to continue . . .
```

7. Supplementary Activity

ANALYSIS:

So in this code, first there's a constant int called size set to 10, which means the array will have 10 elements. Then there's an array named `scores` that holds 10 values, just some random numbers like 95, 85, 78 and so on. After that there's a for loop that prints all the values inside the array using `scores[i]`, just to see what's inside. Then after that, there's another for loop that prints the memory address of each element using `&scores[i]`. This part shows how arrays in c++ are stored in memory like next to each other, in continuous blocks. It helps to understand how memory actually works when using arrays.After that, a pointer called scorePtr is created, and it's set to the address of the first element of the array. then we print out the value of the pointer (which is basically the address of `scores[0]`) and also the value stored in that address by using `*scorePtr`. It shows how pointers can access data from memory. at the end, `sizeof(scores)` is used to find out how much memory the array takes up in total. since it’s 10 integers and usually an int is 4 bytes, the output should be 40 bytes. but that might change depending on the system. Overall, this code just tries to explain arrays, memory and pointers in a simple way.

8. Conclusion

After going through this code and listening to the explanation in our online class earlier, I understood arrays and pointers in c++ much better. I learned that arrays store data in continuous memory locations, which is why it's possible to access their elements using both indexing and pointers. When we printed the addresses using &scores[i], it showed me exactly how memory is laid out for arrays, which was something I never really noticed before.I also learned how pointers can point to specific elements in the array and how we can use the * operator to get the value at that memory address. The part where we used sizeof(scores) taught me how to find out how much memory an array takes up, which can be really useful when working with large data. At first, the pointer part was confusing, but after seeing how it connects to arrays, it started to make sense. This code really helped me understand how memory works in C++ and how powerful pointers can be.