**Batch: C4-1 Roll No.: 16010123217**

**Experiment / assignment / tutorial No. 3**

**Grade: AA / AB / BB / BC / CC / CD /DD**

**Signature of the Staff In-charge with date**

|  |
| --- |
| **TITLE:** Write a program in C to demonstrate use of arrays |

**AIM:** Program to sort the 1D array in the ascending or descending order and then accept the element from user and insert in the same array at its correct place by keeping array sorted

Write a program to find the Transpose of a Matrix.

\_\_\_\_**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Expected OUTCOME of Experiment:**

Apply basic concepts of C programming for problem solving.(CO1 and CO2).

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Books/ Journals/ Websites referred:**

1. Programming in C, second edition, Pradeep Dey and Manas Ghosh, Oxford University Press.
2. Programming in ANSI C, fifth edition, E Balagurusamy, Tata McGraw Hill.
3. Introduction to programming and problem solving , G. Michael Schneider ,Wiley India edition.
4. [**http://cse.iitkgp.ac.in/~rkumar/pds-vlab/**](http://cse.iitkgp.ac.in/~rkumar/pds-vlab/)

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Problem Definition:**

1. The program takes a 1D array and sorts it in the specified manner. The user enters an element and the same has to be inserted at the correct place in the sorted array.
2. Write a program to find the Transpose of a Matrix.

* Entered matrix:

1 4 0

-5 2 7

* Transpose of the matrix:
  1. -5

4 2

0 7

**Algorithm:**

1)Sorted Array with insertion of elements

Step 1: Start

Step 2: Input number of elements (n) from the user

Step 3: Declare an array (arr) of length n

Step 4: Input elements of arr from the user

Step 5: Bubble sort algorithm to sort elements

a. Outer for loop from i=0 to n-1

b. Inner for loop from j=0 to n-1-i

c. If arr[j] > arr[j+1], swap elements

d. After each iteration, the largest element will be at the end of the array

Step 6: Print the sorted array

Step 7: Input the number of elements (a) you wish to insert

Step 8: Create a new array (newArr) of length n + a

Step 9: Copy elements from arr to newArr

Step 10: Input the additional elements for newArr

Step 11: Apply bubble sort to newArr

a. Outer for loop from i=0 to n+a-1

b. Inner for loop from j=0 to n+a-1-i

c. If newArr[j] > newArr[j+1], swap elements

d. After each iteration, the largest element will be at the end of the array

Step 12: Print the sorted array with inserted elements

Step 13: End

2) Transpose of the Matrix

Step 1: Declare integer variables rows, col to store the number of rows and columns.

Step 2: Declare two-dimensional integer arrays og\_arr (original array) and trans\_arr (transposed array) with sizes rows x cols and cols x rows respectively.

Step 3: Display a message asking the user to enter the number of rows.

Step 4: Read the entered number using scanf and store it in rows.

Step 5: Display a message asking the user to enter the number of columns.

Step 6: Read the entered number using scanf and store it in col.

Step 7: Use a nested loop for each row (i) and column (j):

Step 8: Within the inner loop, create a message showing the current row and column index (e.g., "Enter element at [1][2]").

Step 9: Use scanf to read the entered value and store it in og\_arr[i][j].

Step 10: Use a nested loop for each row (i) and column (j):

Step 11: Within the inner loop, copy the value from og\_arr[i][j] to trans\_arr[j][i], effectively swapping the row and column indices.

Step 12: Display a message indicating the start of the original array.

Step 13: Use a nested loop for each row (i) and column (j):

Step 14: Within the inner loop, print the value from og\_arr[i][j] followed by a tab character.

Step 15: After each row, print a newline character to move to the next line.

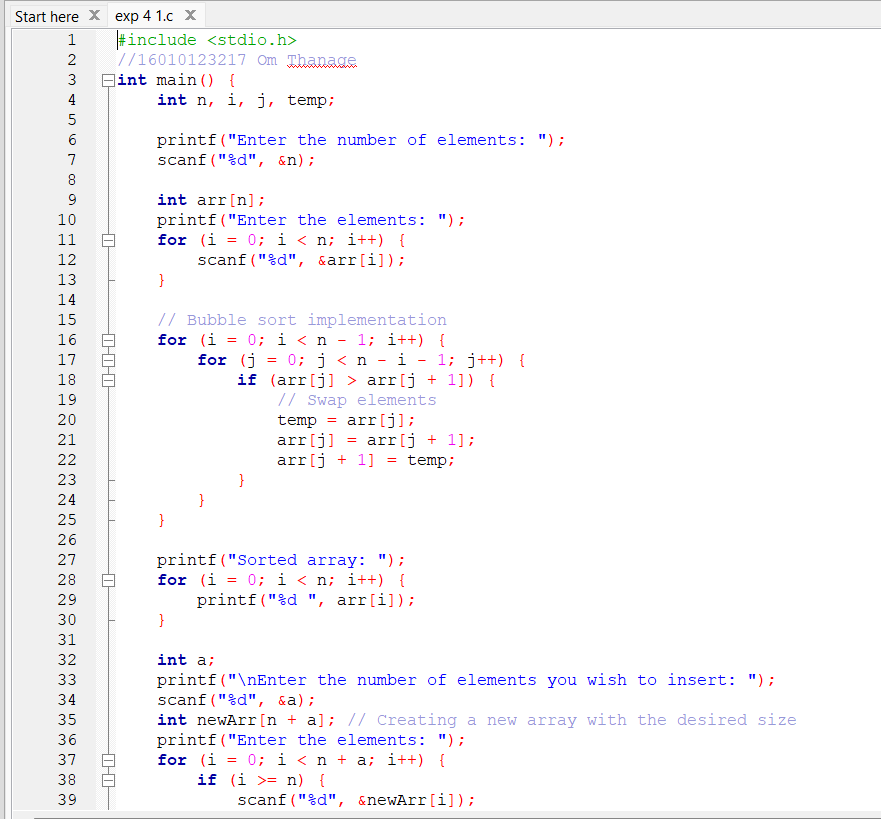
Step 16: Repeat steps 12-15 for the transposed array, displaying a message indicating the start of the transposed array before printing.

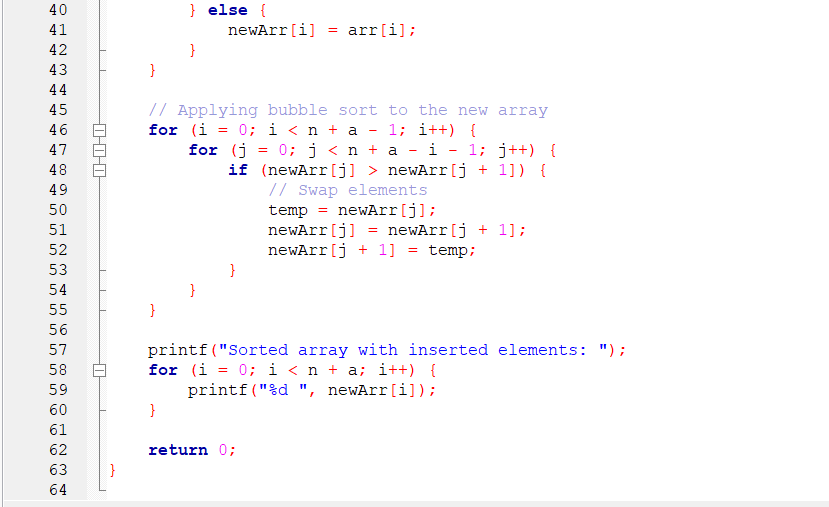
Step 17: Return 0 from main to indicate successful program execution.

**Implementation details:**

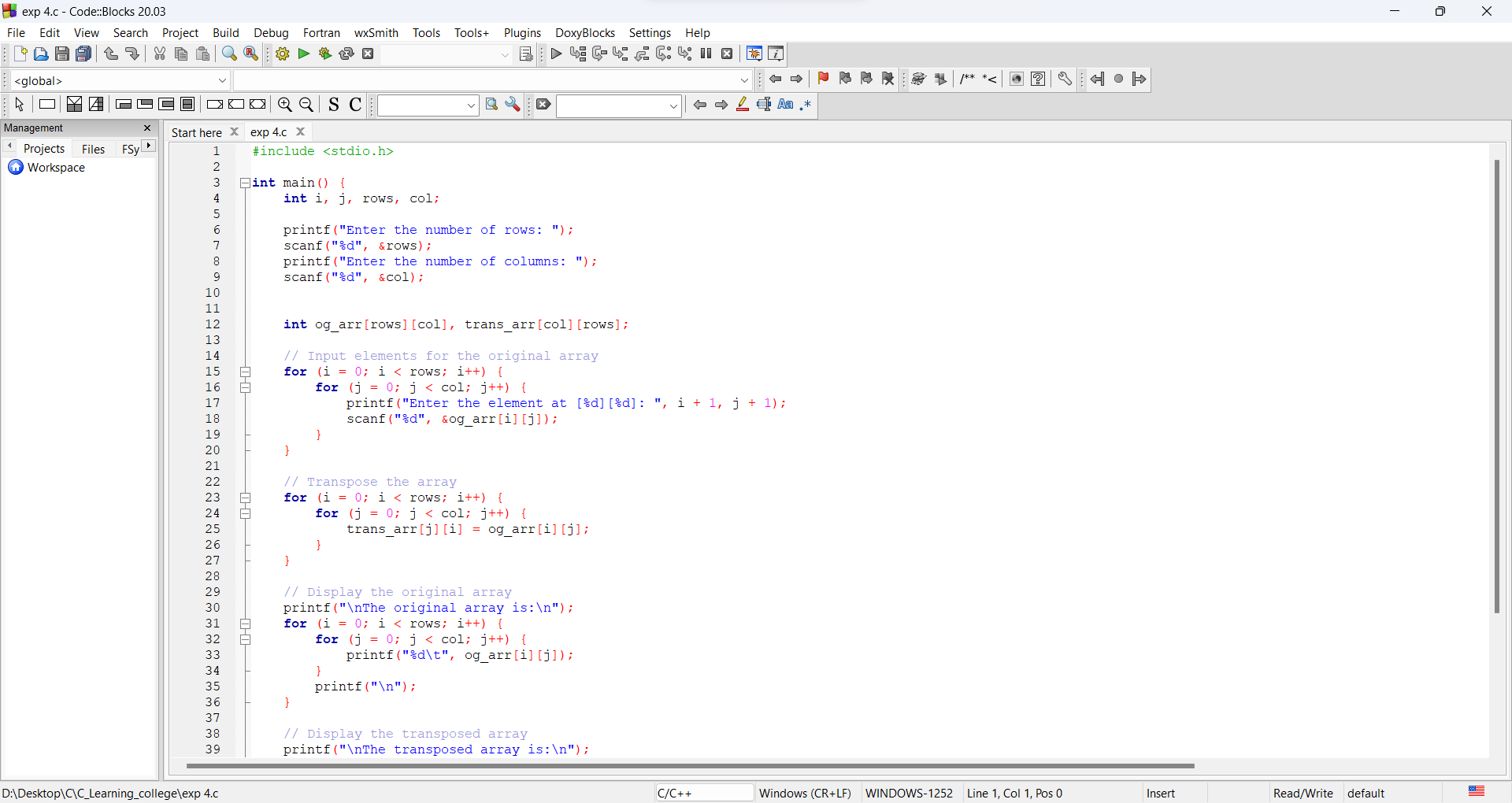
**1) Sorted Array with insertion of elements**

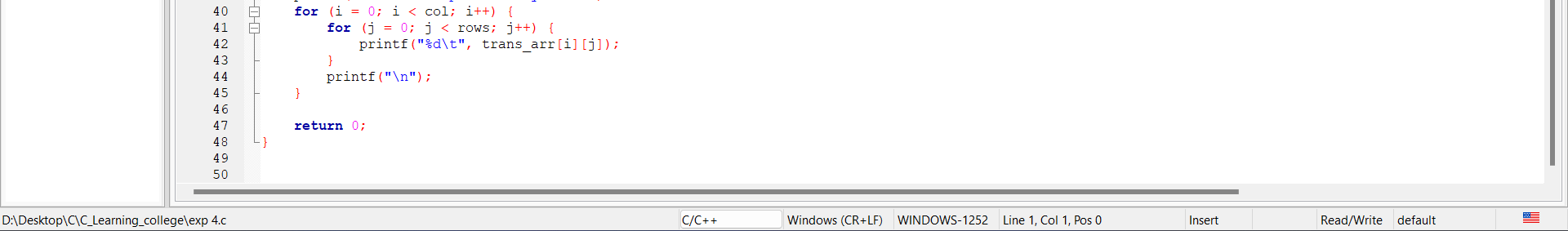
**Code :**





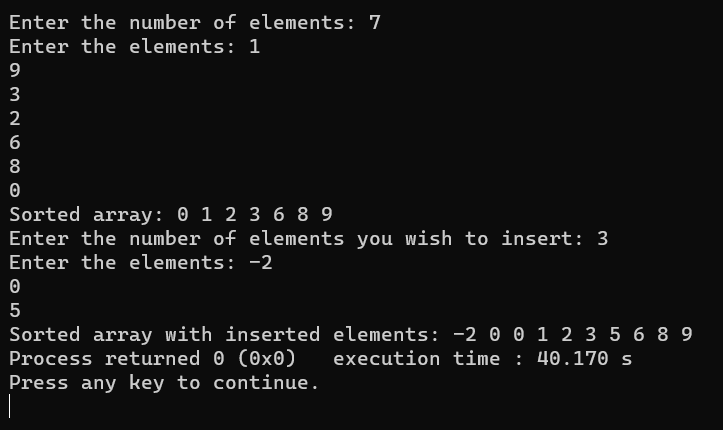
2) Transpose of a Matrix

Code:

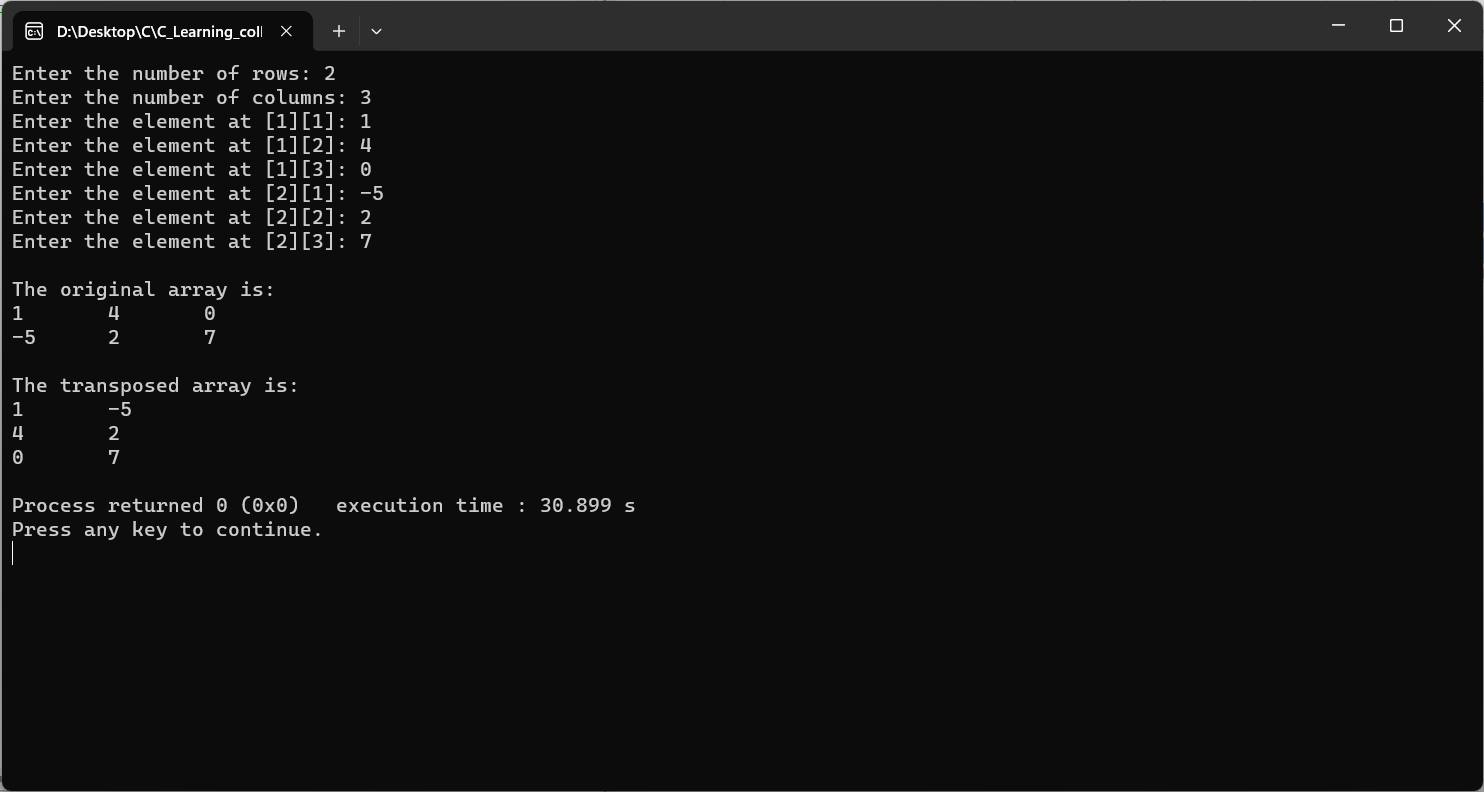


**Output(s):**

**1)Sorted Array with insertion of elements**

****

**2)Transpose of a Matrix**

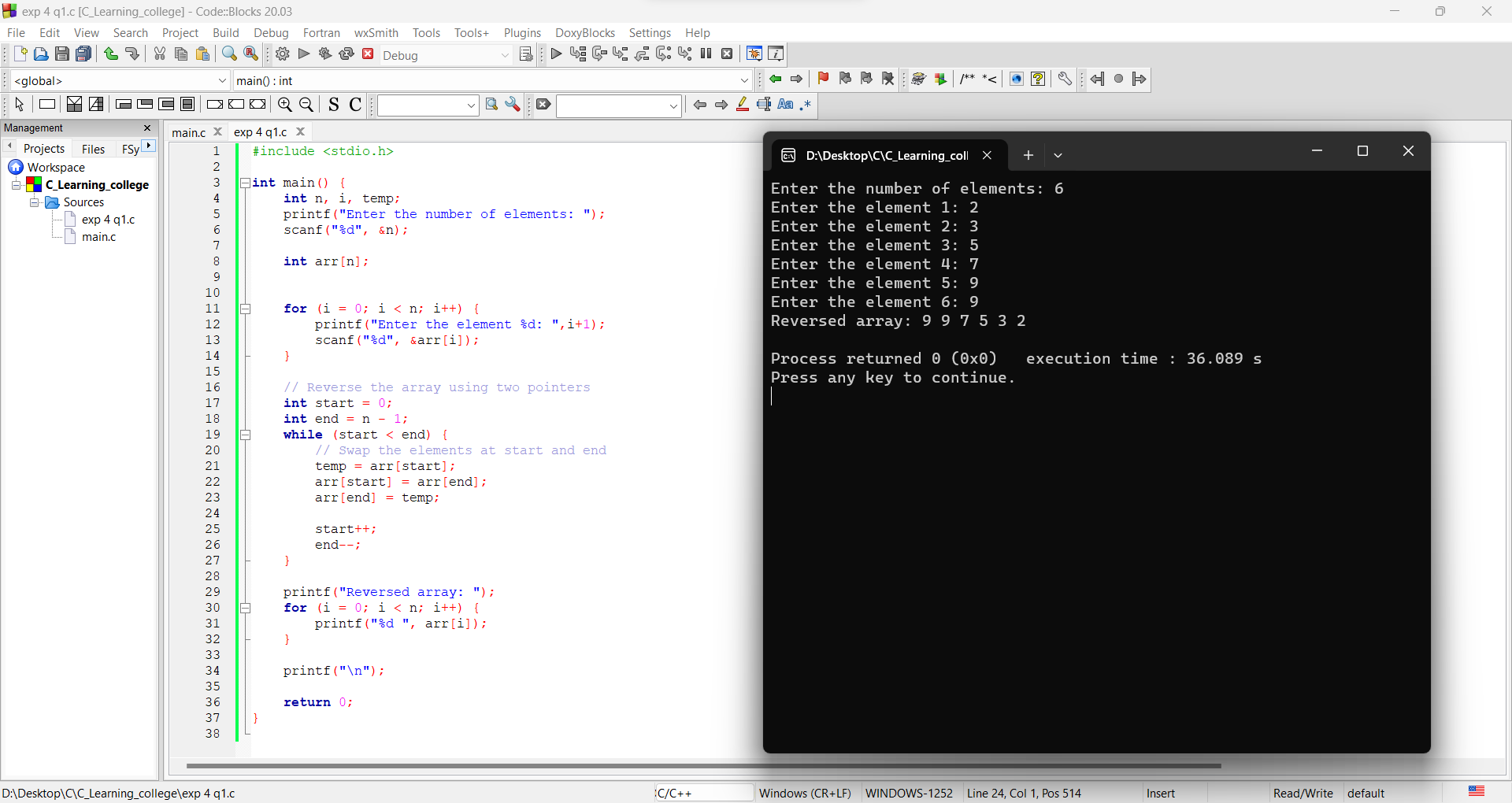


**Conclusion:**

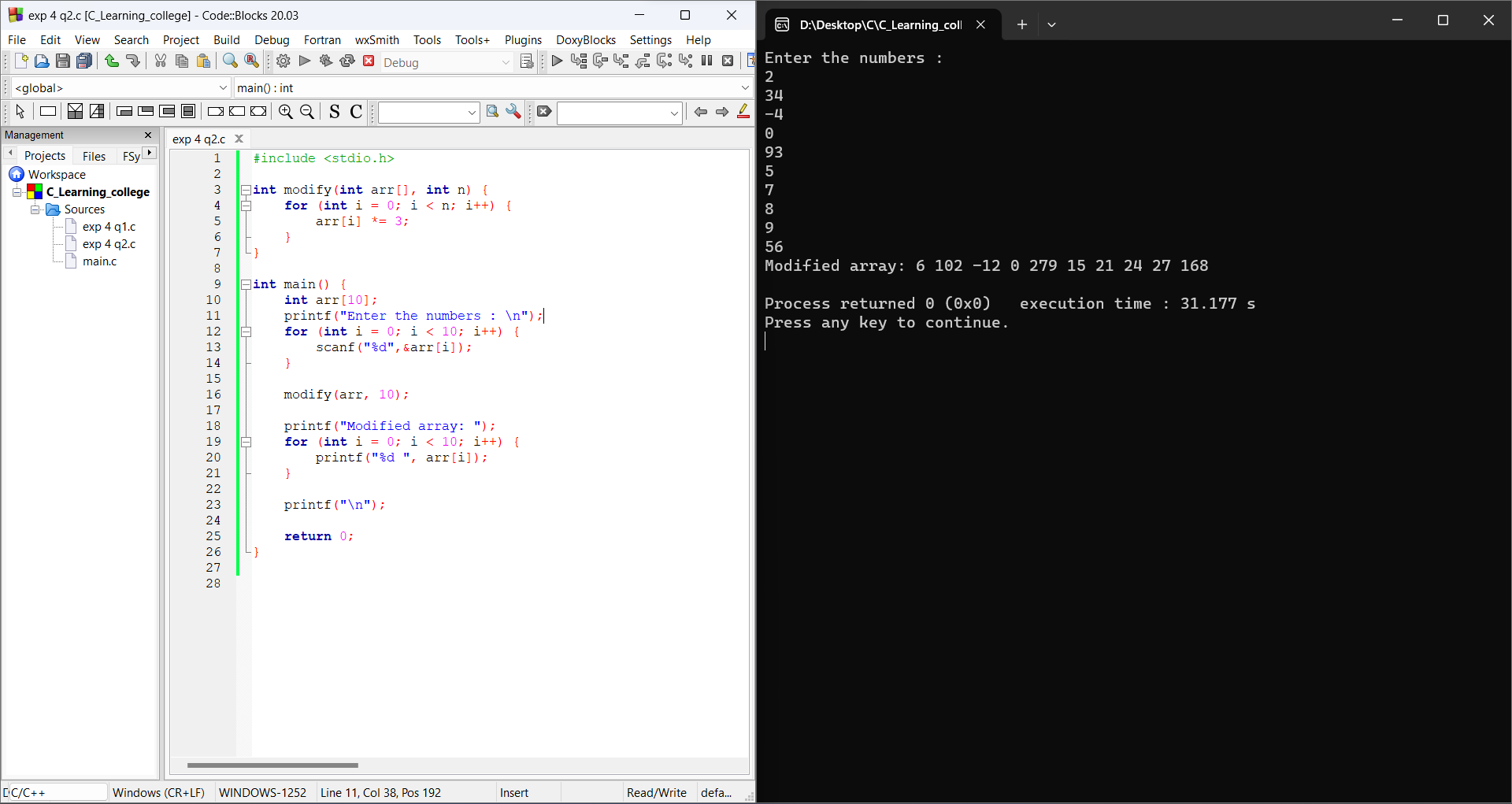
We have learned how to input user-defined arrays in C, as well as how to sort arrays effectively. Additionally, we explored the concept of two-dimensional arrays and delved into finding the transpose of such an array.

**Post Lab Questions**

1. Write a program to enter n numbers, store them in an array and rearrange the array in the reverse order.



1. Write a program which performs the following tasks:
2. Initialize an integer array of 10 elements in main( )
3. Pass the entire array to a function modify( )
4. In modify( ) multiply each element of array by 3
5. Return the control to main( ) and print the new array elements in main( )



**Date: \_\_\_\_\_\_\_\_\_\_\_\_\_ Signature of faculty in-charge**