

In the Name of Allah, the Most Gracious, the Most Merciful

COMSATS UNIVERSITY ISLAMABAD



Department: Computer Science

Submitted By:

ZAIN SALEEM

Submitted To:

MAM YASMEEN JANA

- Course Title : DATA STRUCTURES & ALGORITHMS (LAB)
- Assignment NO : 1
- Registration No : SP22-BCS-126
- Section : B
- Date of submission : 11/09/2023

PROGRAM # 01

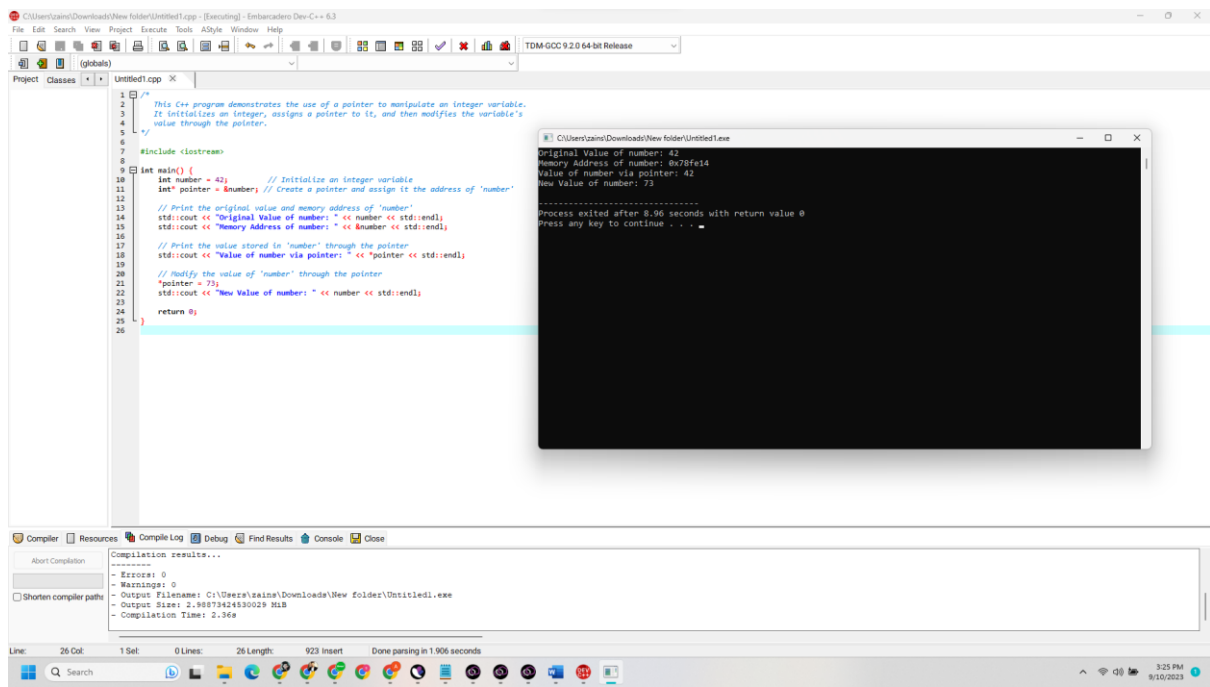
/*

This C++ program demonstrates the use of a pointer to manipulate an integer variable. It initializes an integer, assigns a pointer to it, and then modifies the variable's value through the pointer.

*/

```
#include <iostream>
```

```
int main() {  
    int number = 42;        // Initialize an integer variable  
    int* pointer = &number; // Create a pointer and assign it the address of 'number'  
  
    // Print the original value and memory address of 'number'  
    std::cout << "Original Value of number: " << number << std::endl;  
    std::cout << "Memory Address of number: " << &number << std::endl;  
  
    // Print the value stored in 'number' through the pointer  
    std::cout << "Value of number via pointer: " << *pointer << std::endl;  
  
    // Modify the value of 'number' through the pointer  
    *pointer = 73;  
    std::cout << "New Value of number: " << number << std::endl;  
  
    return 0;  
}
```



PROGRAM # 02

```
/*
  This C++ program calculates the sum of elements in an array
  using a pointer. It prompts the user to enter the size of the array,
  input the array elements, and then calculates the sum using a pointer.
*/
#include <iostream>

int main() {
    int size;

    // Prompt user for the size of the array
    std::cout << "Enter the size of the array: ";
    std::cin >> size;

    // Dynamically allocate memory for the array
    int* arr = new int[size];

    // Prompt user to input array elements
    std::cout << "Enter " << size << " integers:" << std::endl;
    for (int i = 0; i < size; i++) {
        std::cin >> arr[i];
    }

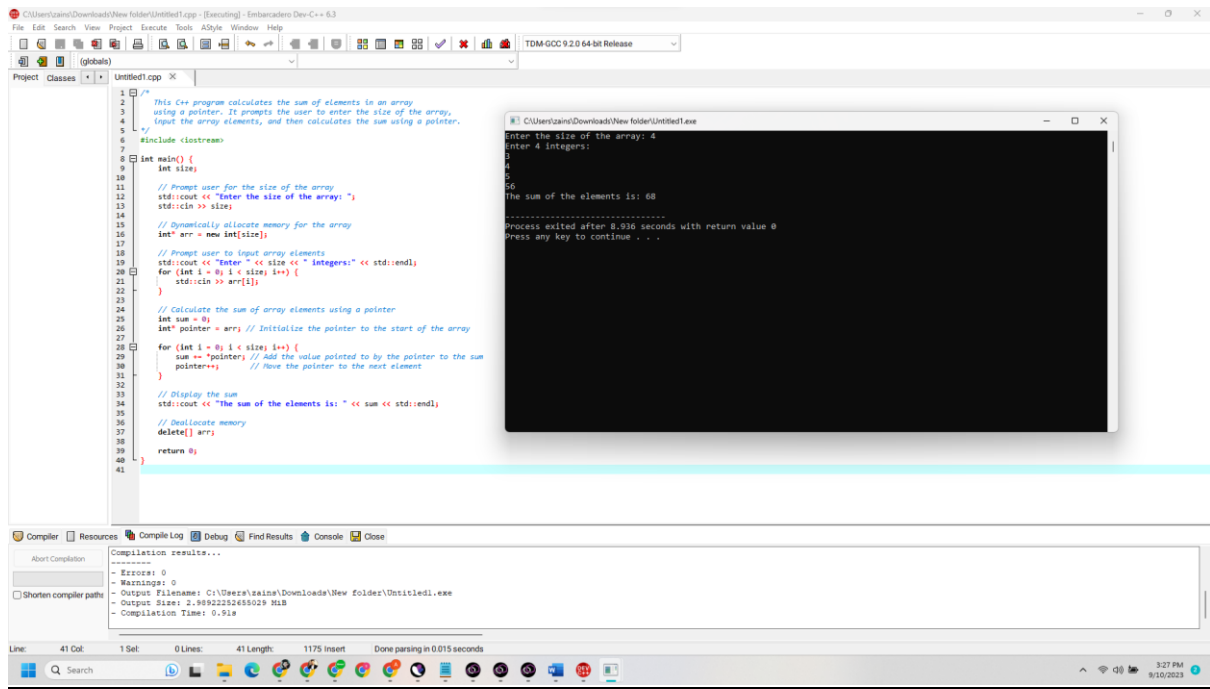
    // Calculate the sum of array elements using a pointer
    int sum = 0;
    int* pointer = arr; // Initialize the pointer to the start of the array

    for (int i = 0; i < size; i++) {
        sum += *pointer; // Add the value pointed to by the pointer to the sum
        pointer++;      // Move the pointer to the next element
    }

    // Display the sum
    std::cout << "The sum of the elements is: " << sum << std::endl;

    // Deallocate memory
    delete[] arr;

    return 0;
}
```



PROGRAM # 03

```
/*
```

This C++ program reverses an array using a pointer.

It prompts the user to enter the size of the array, input the array elements, and then reverses the order of the elements in the array using a pointer.

```
*/
```

```
#include <iostream>
```

```
int main() {
    int size;
```

```
    // Prompt user for the size of the array
    std::cout << "Enter the size of the array: ";
    std::cin >> size;
```

```
    // Dynamically allocate memory for the array
    int* arr = new int[size];
```

```
    // Prompt user to input array elements
    std::cout << "Enter " << size << " integers:" << std::endl;
    for (int i = 0; i < size; i++) {
        std::cin >> arr[i];
    }
```

```

// Reverse the array using a pointer
int* start = arr;           // Pointer to the start of the array
int* end = arr + size - 1; // Pointer to the end of the array

while (start < end) {
    // Swap the elements pointed to by start and end
    int temp = *start;
    *start = *end;
    *end = temp;

    // Move the pointers toward each other
    start++;
    end--;
}

// Display the reversed array
std::cout << "Reversed array: ";
for (int i = 0; i < size; i++) {
    std::cout << arr[i] << " ";
}
std::cout << std::endl;

// Deallocate memory
delete[] arr;

return 0;
}

```

The screenshot displays a C++ program in an IDE, specifically using the TDM-GCC 9.2.0 64-bit Release compiler. The code implements a function to reverse an array using pointers. It prompts the user for the array size and elements, then reverses the array in-place by swapping elements from both ends towards the center. Finally, it displays the reversed array and deallocates the memory.

The execution output shows the following interaction:

```

Enter the size of the array: 2
Enter 2 integers:
3
4
Reversed array: 4 3
.....
Process exited after 14.43 seconds with return value 0
Press any key to continue . . .

```

The IDE interface includes a menu bar (File, Edit, Search, View, Project, Execute, Tools, AStyle, Window, Help), a toolbar, and a status bar at the bottom showing line and column information (Line: 51 Col: 1, 1 Sel: 0 Lines: 51 Length: 1355 Insert: Done parsing in 0.047 seconds).

PROGRAM # 04

```
/*
  This C++ program calculates the length of a C-style string (null-terminated string)
  using a pointer. It prompts the user to enter a string, and then calculates and
  displays the length of the string using a pointer.
*/
#include <iostream>

int main() {
    const int maxStringLength = 100; // Maximum string length

    char str[maxStringLength]; // Declare a character array to store the string

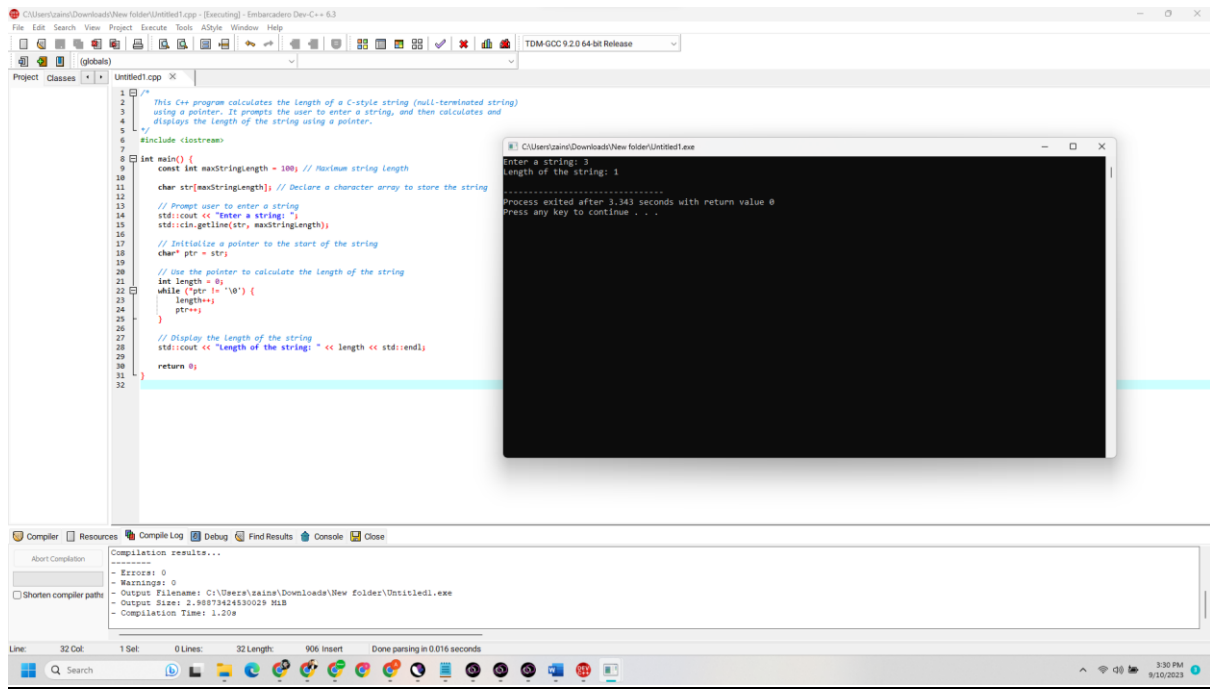
    // Prompt user to enter a string
    std::cout << "Enter a string: ";
    std::cin.getline(str, maxStringLength);

    // Initialize a pointer to the start of the string
    char* ptr = str;

    // Use the pointer to calculate the length of the string
    int length = 0;
    while (*ptr != '\0') {
        length++;
        ptr++;
    }

    // Display the length of the string
    std::cout << "Length of the string: " << length << std::endl;

    return 0;
}
```



PROGRAM # 05

/*

This C++ program demonstrates the use of pointers to swap two numbers. It prompts the user to input two integers, swaps their values using a pointer, and then displays the swapped values.

*/

```
#include <iostream>
```

```
int main() {
    int num1, num2;
```

```
    // Prompt user for input
    std::cout << "Enter the first number: ";
    std::cin >> num1;
    std::cout << "Enter the second number: ";
    std::cin >> num2;
```

```
    // Declare a pointer to int
    int* pointer;
```

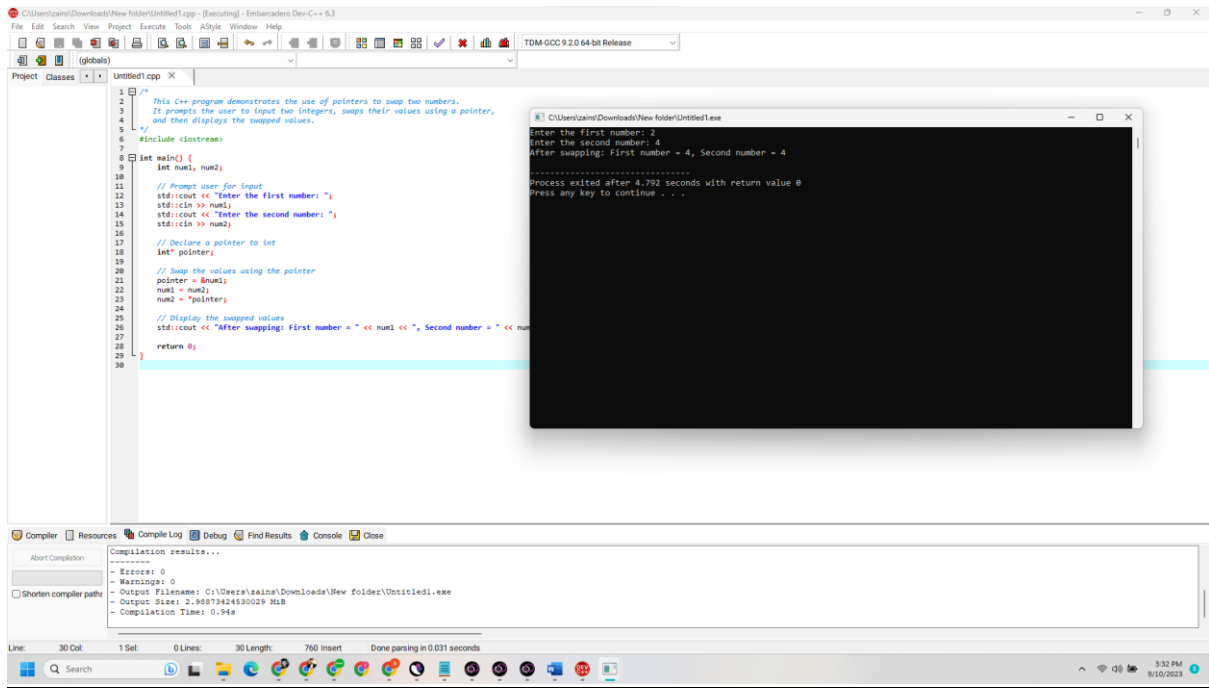
```
    // Swap the values using the pointer
    pointer = &num1;
    num1 = num2;
    num2 = *pointer;
```

```

// Display the swapped values
std::cout << "After swapping: First number = " << num1 << ", Second number = " <<
num2 << std::endl;

return 0;
}

```



PROGRAM # 06

```
/*
    This C++ program calculates the sum of elements in an array using a pointer.
    It prompts the user to input the size of the array, the array elements,
    and then calculates and displays the sum of the elements using a pointer.
*/
#include <iostream>

int main() {
    int size;

    // Prompt user for the size of the array
    std::cout << "Enter the size of the array: ";
    std::cin >> size;

    // Dynamically allocate memory for the array
    int* arr = new int[size];

    // Prompt user to input array elements
    std::cout << "Enter " << size << " integers:" << std::endl;
    for (int i = 0; i < size; i++) {
        std::cin >> arr[i];
    }

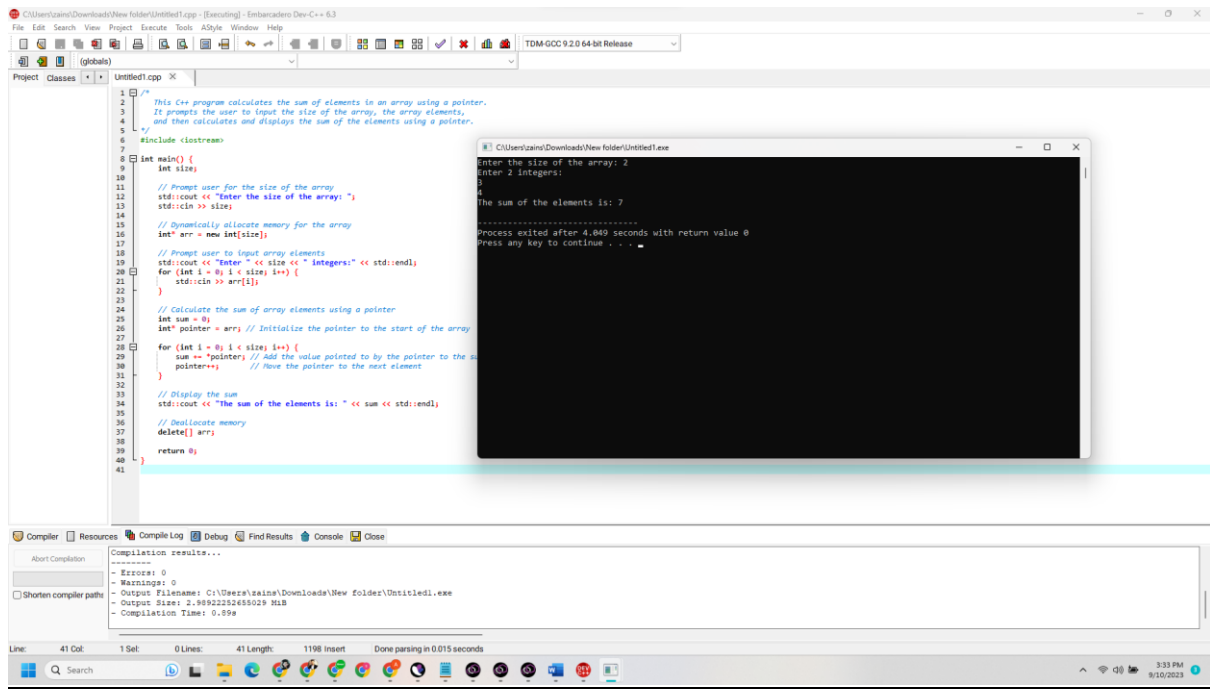
    // Calculate the sum of array elements using a pointer
    int sum = 0;
    int* pointer = arr; // Initialize the pointer to the start of the array

    for (int i = 0; i < size; i++) {
        sum += *pointer; // Add the value pointed to by the pointer to the sum
        pointer++;      // Move the pointer to the next element
    }

    // Display the sum
    std::cout << "The sum of the elements is: " << sum << std::endl;

    // Deallocate memory
    delete[] arr;

    return 0;
}
```



PROGRAM # 07

/*

This C++ program calculates the factorial of a non-negative integer using a pointer. It prompts the user to input a non-negative integer, calculates its factorial using a pointer, and then displays the result.

*/

```
#include <iostream>
```

```
int main() {
    int n;
```

```
    // Prompt user for input
    std::cout << "Enter a non-negative integer: ";
    std::cin >> n;
```

```
    if (n < 0) {
        std::cout << "Factorial is not defined for negative numbers." << std::endl;
    } else {
        int factorial = 1;
        int* ptrFactorial = &factorial;
```

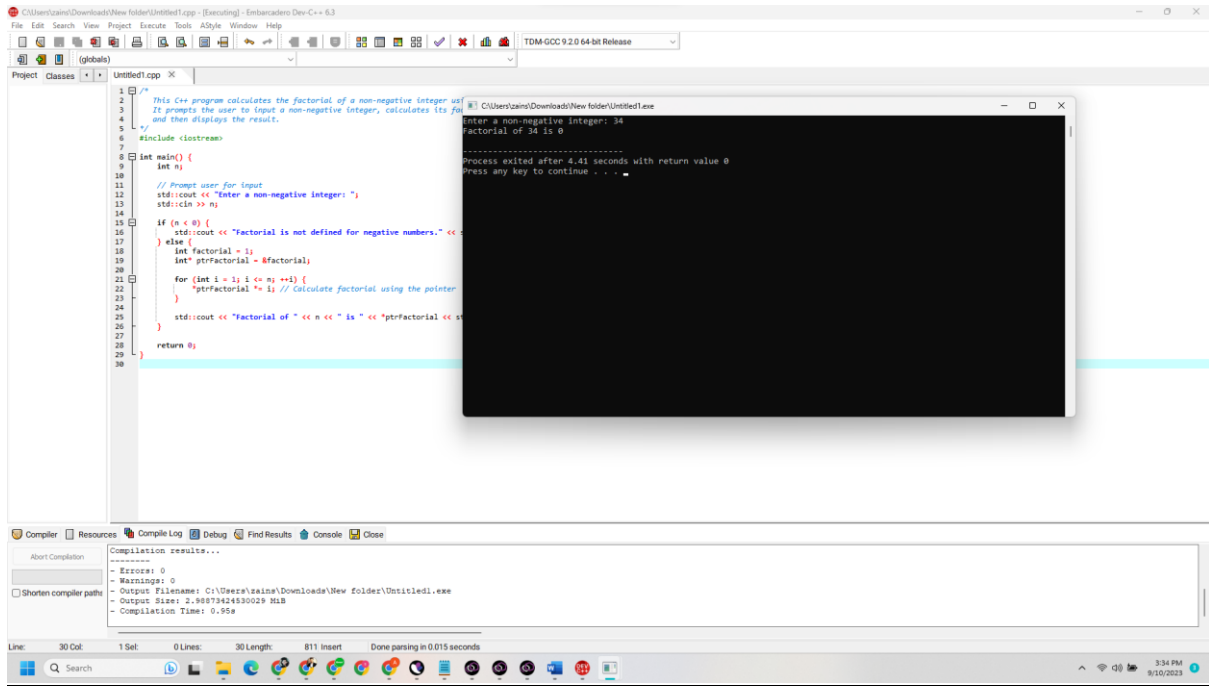
```
        for (int i = 1; i <= n; ++i) {
            *ptrFactorial *= i; // Calculate factorial using the pointer
        }
    }
}
```

```

        std::cout << "Factorial of " << n << " is " << *ptrFactorial << std::endl;
    }

    return 0;
}

```



PROGRAM # 08

```

/*
This C++ program reverses a string using a pointer.
It prompts the user to input a string, reverses it using a pointer,
and then displays the reversed string.
*/
#include <iostream>
#include <cstring>

int main() {
    const int maxStringLength = 100; // Maximum string length

    char str[maxStringLength]; // Declare a character array to store the string

    // Prompt user to enter a string
    std::cout << "Enter a string: ";
    std::getline(str, maxStringLength);

```

```

// Use pointers to reverse the string
char* start = str;
char* end = str + std::strlen(str) - 1;

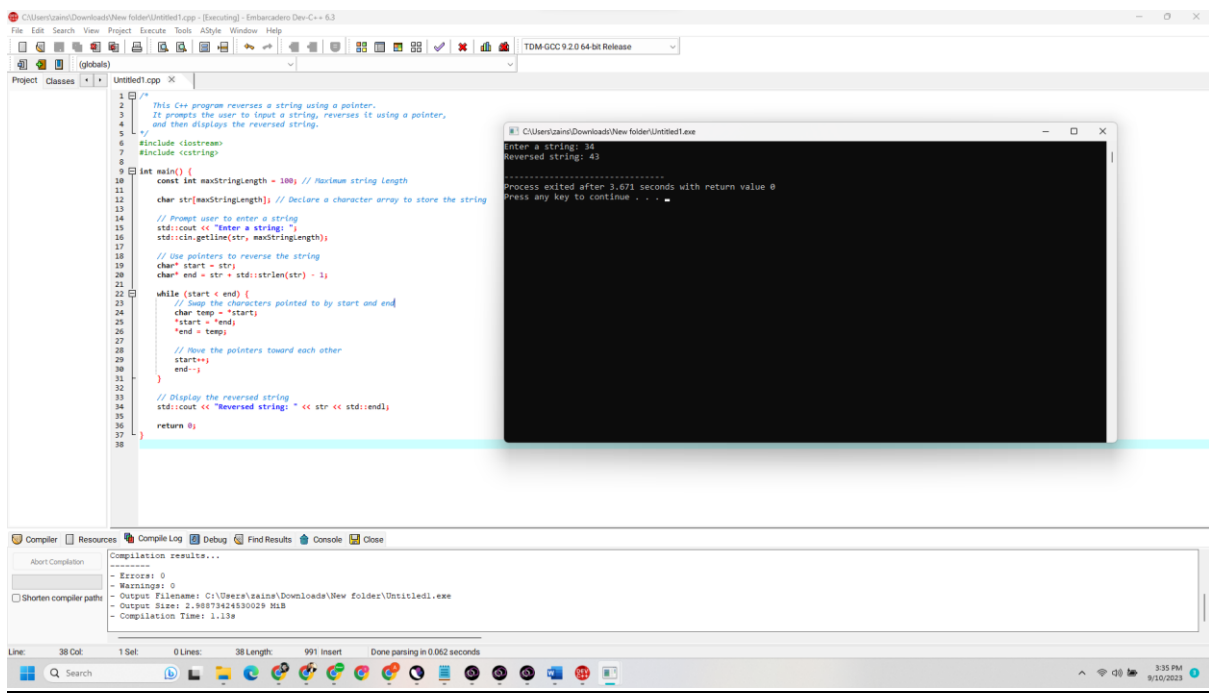
while (start < end) {
    // Swap the characters pointed to by start and end
    char temp = *start;
    *start = *end;
    *end = temp;

    // Move the pointers toward each other
    start++;
    end--;
}

// Display the reversed string
std::cout << "Reversed string: " << str << std::endl;

return 0;
}

```



PROGRAM # 09

```
/*
  This C++ program calculates the square of each element in an array using a pointer.
  It prompts the user to input the size of the array, the array elements,
  calculates the squares using a pointer, and displays the squared values.
*/
#include <iostream>

int main() {
    int size;

    // Prompt user for the size of the array
    std::cout << "Enter the size of the array: ";
    std::cin >> size;

    // Dynamically allocate memory for the array
    int* arr = new int[size];

    // Prompt user to input array elements
    std::cout << "Enter " << size << " integers:" << std::endl;
    for (int i = 0; i < size; i++) {
        std::cin >> arr[i];
    }

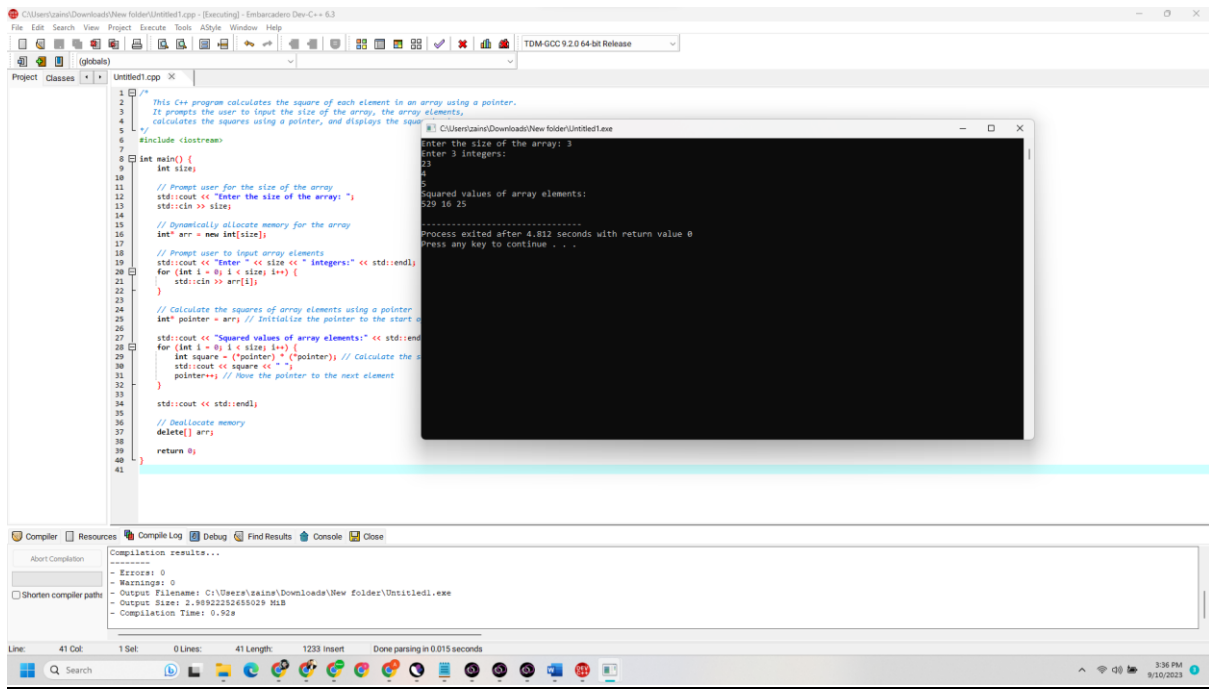
    // Calculate the squares of array elements using a pointer
    int* pointer = arr; // Initialize the pointer to the start of the array

    std::cout << "Squared values of array elements:" << std::endl;
    for (int i = 0; i < size; i++) {
        int square = (*pointer) * (*pointer); // Calculate the square using the pointer
        std::cout << square << " ";
        pointer++; // Move the pointer to the next element
    }

    std::cout << std::endl;

    // Deallocate memory
    delete[] arr;

    return 0;
}
```



PROGRAM # 10

/*

This C++ program calculates the average of elements in an array using a pointer. It prompts the user to input the size of the array, the array elements, calculates the average using a pointer, and displays the result.

*/

```
#include <iostream>
```

```
int main() {
    int size;
```

```
    // Prompt user for the size of the array
    std::cout << "Enter the size of the array: ";
    std::cin >> size;
```

```
    // Dynamically allocate memory for the array
    int* arr = new int[size];
```

```
    // Prompt user to input array elements
    std::cout << "Enter " << size << " integers:" << std::endl;
    for (int i = 0; i < size; i++) {
        std::cin >> arr[i];
    }
```

```
    // Calculate the average of array elements using a pointer
    int sum = 0;
```

```
int* pointer = arr; // Initialize the pointer to the start of the array
```

```
for (int i = 0; i < size; i++) {  
    sum += *pointer; // Add the value pointed to by the pointer to the sum  
    pointer++;      // Move the pointer to the next element  
}
```

```
double average = static_cast<double>(sum) / size; // Calculate the average
```

```
// Display the average
```

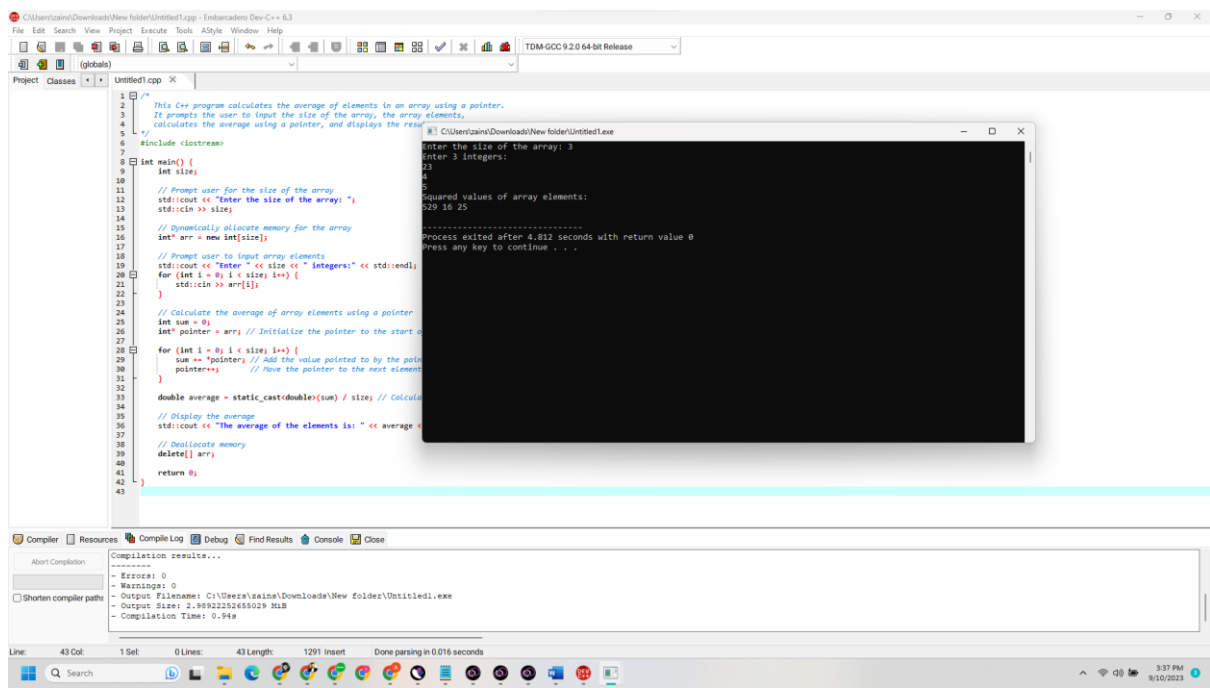
```
std::cout << "The average of the elements is: " << average << std::endl;
```

```
// Deallocate memory
```

```
delete[] arr;
```

```
return 0;
```

```
}
```



The screenshot shows a C++ IDE with the following components:

- Editor:** Displays the C++ code for calculating the average of array elements using a pointer. The code includes comments and standard C++ syntax.
- Output Window:** Shows the execution results, including the input size (3), the input integers (5, 29, 16), the squared values (25, 841, 256), and the final average (17.5).
- Compiler Output:** Shows the compilation results, indicating that the program compiled successfully without errors or warnings.

```
1  /* This C++ program calculates the average of elements in an array using a pointer.  
2  3  4  It prompts the user to input the size of the array, the array elements,  
5  6  7  calculates the average using a pointer, and displays the result.  
8  9  10 #include <iostream>  
11 12 13 int main() {  
14 15     int size;  
16 17     // Prompt user for the size of the array  
18 19     std::cout << "Enter the size of the array: ";  
20 21     std::cin >> size;  
22 23     // Dynamically allocate memory for the array  
24 25     int* arr = new int[size];  
26 27     // Prompt user to input array elements  
28 29     std::cout << "Enter " << size << " integers: " << std::endl;  
30 31     for (int i = 0; i < size; i++) {  
32 33         std::cin >> arr[i];  
34 35     }  
36 37     // Calculate the average of array elements using a pointer  
38 39     int sum = 0;  
40 41     int* pointer = arr; // Initialize the pointer to the start of the array  
42 43     for (int i = 0; i < size; i++) {  
44 45         sum += *pointer; // Add the value pointed to by the pointer to the sum  
46 47         pointer++;      // Move the pointer to the next element  
48 49     }  
49 50     double average = static_cast<double>(sum) / size; // Calculate the average  
51 52     // Display the average  
53 54     std::cout << "The average of the elements is: " << average << std::endl;  
55 56     // Deallocate memory  
57 58     delete[] arr;  
59 60     return 0;  
61 62 }
```

Output:

```
Enter the size of the array: 3  
Enter 3 integers:  
5  
29  
16  
Squared values of array elements:  
25  
841  
256  
Process exited after 4.812 seconds with return value 0  
Press any key to continue . . .
```

Compilation results:

```
Errors: 0  
Warnings: 0  
Output Filename: C:\Users\jains\Downloads\New Folder\Untitled1.exe  
Output Size: 2.9892232455029 MB  
Compilation Time: 0.94s
```

PROGRAM # 11

```
/*
  This C++ program finds the largest element in an array using a pointer.
  It prompts the user to input the size of the array, the array elements,
  calculates the maximum using a pointer, and displays the largest element.
*/
#include <iostream>

int main() {
    int size;

    // Prompt user for the size of the array
    std::cout << "Enter the size of the array: ";
    std::cin >> size;

    // Dynamically allocate memory for the array
    int* arr = new int[size];

    // Prompt user to input array elements
    std::cout << "Enter " << size << " integers:" << std::endl;
    for (int i = 0; i < size; i++) {
        std::cin >> arr[i];
    }

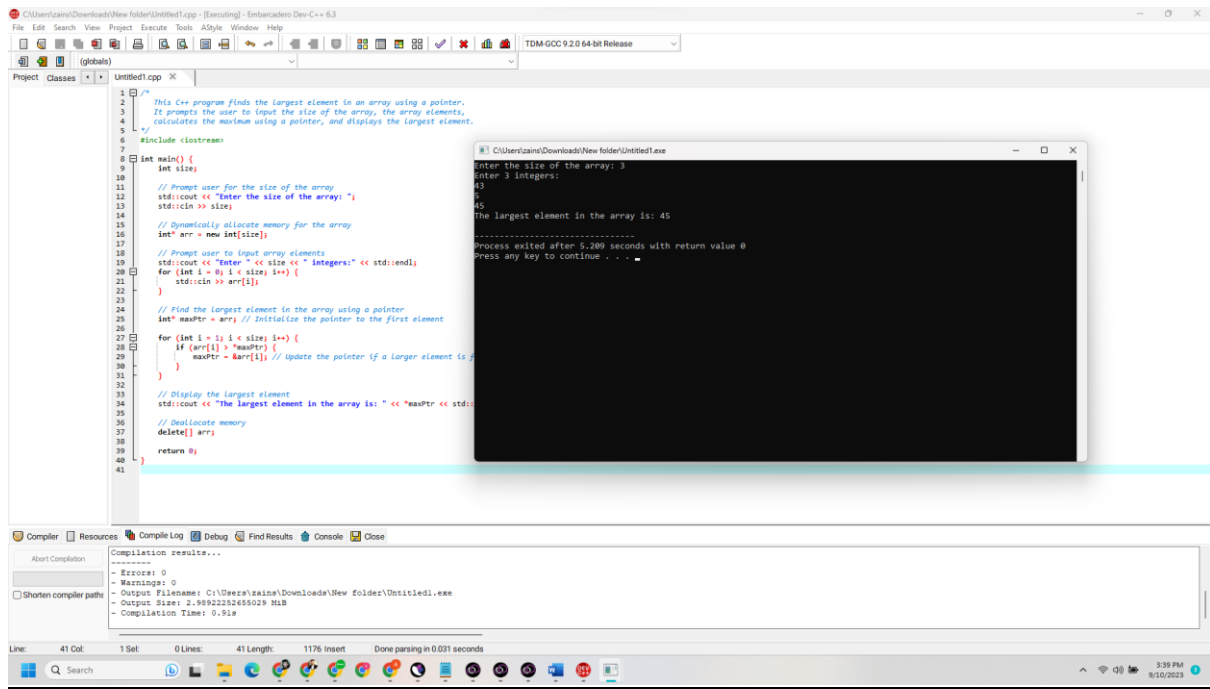
    // Find the largest element in the array using a pointer
    int* maxPtr = arr; // Initialize the pointer to the first element

    for (int i = 1; i < size; i++) {
        if (arr[i] > *maxPtr) {
            maxPtr = &arr[i]; // Update the pointer if a larger element is found
        }
    }

    // Display the largest element
    std::cout << "The largest element in the array is: " << *maxPtr << std::endl;

    // Deallocate memory
    delete[] arr;

    return 0;
}
```

PROGRAM # 12

/*

This C++ program performs a linear search in an array to find a specific element using a pointer.

It prompts the user to input the size of the array, the array elements, and the element to search for. It uses a pointer to perform the search and displays whether the element was found and its position if present.

*/

#include <iostream>

```
int main() {
    int size;
```

```
    // Prompt user for the size of the array
    std::cout << "Enter the size of the array: ";
    std::cin >> size;
```

```
    // Dynamically allocate memory for the array
    int* arr = new int[size];
```

```
    // Prompt user to input array elements
    std::cout << "Enter " << size << " integers:" << std::endl;
    for (int i = 0; i < size; i++) {
        std::cin >> arr[i];
    }
```

```
    int target;
    bool found = false;
```

```
int position = -1;
```

```
// Prompt user to input the element to search for
std::cout << "Enter the element to search for: ";
std::cin >> target;
```

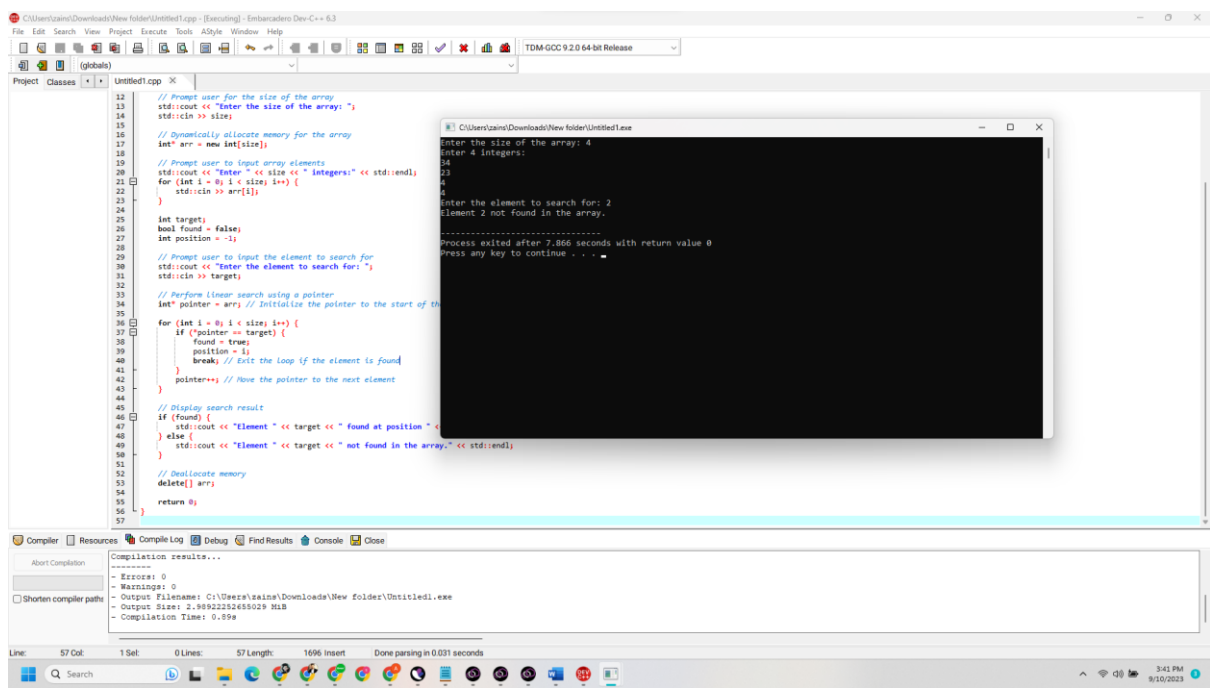
```
// Perform linear search using a pointer
int* pointer = arr; // Initialize the pointer to the start of the array
```

```
for (int i = 0; i < size; i++) {
    if (*pointer == target) {
        found = true;
        position = i;
        break; // Exit the loop if the element is found
    }
    pointer++; // Move the pointer to the next element
}
```

```
// Display search result
if (found) {
    std::cout << "Element " << target << " found at position " << position << std::endl;
} else {
    std::cout << "Element " << target << " not found in the array." << std::endl;
}
```

```
// Deallocate memory
delete[] arr;
```

```
return 0;
}
```



PROGRAM # 13

/*

This C++ program performs a binary search in a sorted array to find a specific element using a pointer.

It prompts the user to input the size of the array, the sorted array elements, and the element to search for. It uses a pointer to perform the binary search and displays whether the element was found and its position if present.

*/

```
#include <iostream>
```

```
int main() {
```

```
    int size;
```

```
    // Prompt user for the size of the array
```

```
    std::cout << "Enter the size of the sorted array: ";
```

```
    std::cin >> size;
```

```
    // Dynamically allocate memory for the sorted array
```

```
    int* arr = new int[size];
```

```
    // Prompt user to input sorted array elements
```

```
    std::cout << "Enter " << size << " sorted integers:" << std::endl;
```

```
    for (int i = 0; i < size; i++) {
```

```
        std::cin >> arr[i];
```

```
    }
```

```
    int target;
```

```
    bool found = false;
```

```
    int position = -1;
```

```
    // Prompt user to input the element to search for
```

```
    std::cout << "Enter the element to search for: ";
```

```
    std::cin >> target;
```

```
    // Perform binary search using a pointer
```

```
    int* left = arr; // Initialize the left pointer to the start of the array
```

```
    int* right = arr + size - 1; // Initialize the right pointer to the end of the array
```

```
    while (left <= right) {
```

```
        int* mid = left + (right - left) / 2; // Calculate the middle pointer
```

```
        if (*mid == target) {
```

```
            found = true;
```

```
            position = mid - arr; // Calculate the position based on pointer subtraction
```

```
            break;
```

```
        } else if (*mid < target) {
```

```
            left = mid + 1; // Adjust the left pointer
```

```
        } else {
```

```
            right = mid - 1; // Adjust the right pointer
```

```

    }
}

// Display search result
if (found) {
    std::cout << "Element " << target << " found at position " << position << std::endl;
} else {
    std::cout << "Element " << target << " not found in the array." << std::endl;
}

// Deallocate memory
delete[] arr;

return 0;
}

```

The screenshot shows a C++ IDE with the following code in `Untitled1.cpp`:

```

18 // Prompt user to input sorted array elements
19 std::cout << "Enter -<< size << " sorted integers:" << std::endl;
20 for (int i = 0; i < size; i++) {
21     std::cin >> arr[i];
22 }
23
24
25 int target;
26 bool found = false;
27 int position = -1;
28
29 // Prompt user to input the element to search for
30 std::cout << "Enter the element to search for: ";
31 std::cin >> target;
32
33 // Perform binary search using a pointer
34 int* left = arr; // Initialize the left pointer to the start of
35 int* right = arr + size - 1; // Initialize the right pointer to
36
37 while (left <= right) {
38     int* mid = left + (right - left) / 2; // Calculate the middle
39
40     if (*mid == target) {
41         found = true;
42         position = mid - arr; // Calculate the position based on
43         break;
44     } else if (*mid < target) {
45         left = mid + 1; // Adjust the left pointer
46     } else {
47         right = mid - 1; // Adjust the right pointer
48     }
49 }
50
51 // Display search result
52 if (found) {
53     std::cout << "Element " << target << " found at position "
54 } else {
55     std::cout << "Element " << target << " not found in the array." << std::endl;
56 }
57
58 // Deallocate memory
59 delete[] arr;
60
61 return 0;
62
63

```

The console output shows the following interaction:

```

C:\Users\zain\Downloads\New folder\Untitled1.exe
Enter the size of the sorted array: 2
Enter 2 sorted integers:
32
43
Enter the element to search for: 3
Element 3 not found in the array.
-----
Process exited after 7.939 seconds with return value 0
Press any key to continue . . .

```

The bottom panel shows the compilation results:

```

Compilation results...
-----
- Errors: 0
- Warnings: 0
- Output Filename: C:\Users\zain\Downloads\New folder\Untitled1.exe
- Output Size: 2,989,223,528,550,29 HkB
- Compilation Time: 0.55s

```

PROGRAM # 14

/*

This C++ program counts the number of occurrences of a specific element in an array using a pointer.

It prompts the user to input the size of the array, the array elements, and the element to count. It uses a pointer to traverse the array and count occurrences.

*/

```
#include <iostream>
```

```
int main() {  
    int size;
```

```
    // Prompt user for the size of the array  
    std::cout << "Enter the size of the array: ";  
    std::cin >> size;
```

```
    // Dynamically allocate memory for the array  
    int* arr = new int[size];
```

```
    // Prompt user to input array elements  
    std::cout << "Enter " << size << " integers:" << std::endl;  
    for (int i = 0; i < size; i++) {  
        std::cin >> arr[i];  
    }
```

```
    int target;  
    int count = 0;
```

```
    // Prompt user to input the element to count  
    std::cout << "Enter the element to count: ";  
    std::cin >> target;
```

```
    // Use a pointer to traverse the array and count occurrences  
    int* pointer = arr; // Initialize the pointer to the start of the array
```

```
    for (int i = 0; i < size; i++) {  
        if (*pointer == target) {  
            count++;  
        }  
        pointer++; // Move the pointer to the next element  
    }
```

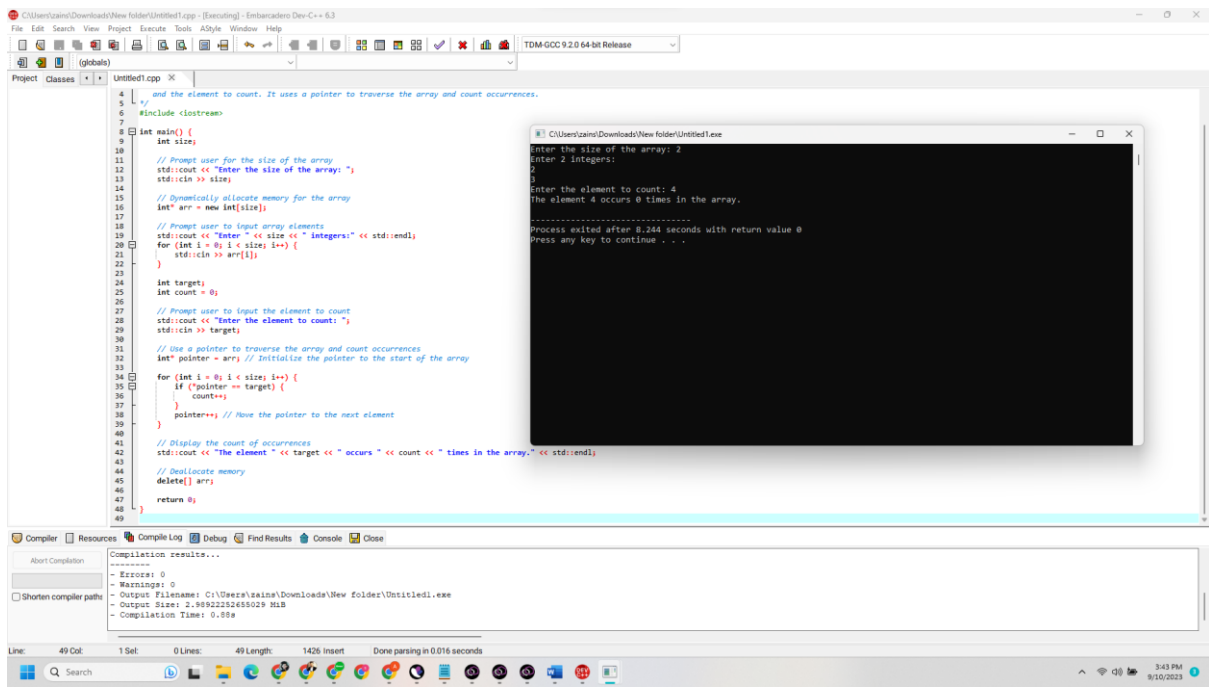
```
    // Display the count of occurrences  
    std::cout << "The element " << target << " occurs " << count << " times in the array." <<  
    std::endl;
```

```
    // Deallocate memory  
    delete[] arr;
```

```

return 0;
}

```



PROGRAM # 15

```

/*

```

This C++ program calculates the transpose of a matrix using pointers. It prompts the user to input the dimensions of a matrix and its elements, and then calculates the transpose using pointers. It displays the transpose.

```

*/

```

```

#include <iostream>

```

```

int main() {
    int rows, cols;

```

```

// Prompt user for the dimensions of the matrix
std::cout << "Enter the number of rows and columns for the matrix: ";
std::cin >> rows >> cols;

```

```

// Dynamically allocate memory for the matrix
int** matrix = new int*[rows];
for (int i = 0; i < rows; i++) {
    matrix[i] = new int[cols];
}

```

```

// Prompt user to input elements of the matrix
std::cout << "Enter elements of the matrix:" << std::endl;
for (int i = 0; i < rows; i++) {

```

```

        for (int j = 0; j < cols; j++) {
            std::cin >> matrix[i][j];
        }
    }

    // Calculate the transpose of the matrix and store it in a new matrix
    int** transpose = new int*[cols];
    for (int i = 0; i < cols; i++) {
        transpose[i] = new int[rows];
    }

    for (int i = 0; i < rows; i++) {
        for (int j = 0; j < cols; j++) {
            transpose[j][i] = matrix[i][j];
        }
    }

    // Display the transpose of the matrix
    std::cout << "Transpose of the matrix:" << std::endl;
    for (int i = 0; i < cols; i++) {
        for (int j = 0; j < rows; j++) {
            std::cout << transpose[i][j] << " ";
        }
        std::cout << std::endl;
    }

    // Deallocate memory for matrices
    for (int i = 0; i < rows; i++) {
        delete[] matrix[i];
    }
    delete[] matrix;

    for (int i = 0; i < cols; i++) {
        delete[] transpose[i];
    }
    delete[] transpose;

    return 0;
}

```

