

# Assignment - 1

## C++

**Name:** Kamithkar Vinod

**Course:** PG DAC AUGUST 2025

**PRN:** 250850320040

**Form No:** 250500480

**Date:** 07-10-2025

---

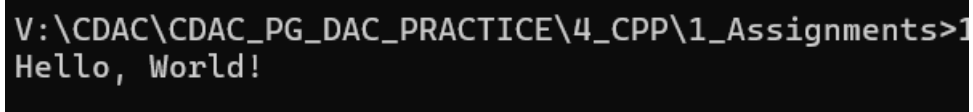
### Problem 1:

**Task:** Write a C++ program that prints "Hello, World!" to the console.

**Code:** —

```
1 #include <iostream>
2 using namespace std;
3
4 int main() {
5     cout << "Hello, World!" << endl;
6     return 0;
7 }
```

**Output: 1**—



```
V:\CDAC\CDAC_PG_DAC_PRACTICE\4_CPP\1_Assignments>1
Hello, World!
```

### Problem 2:

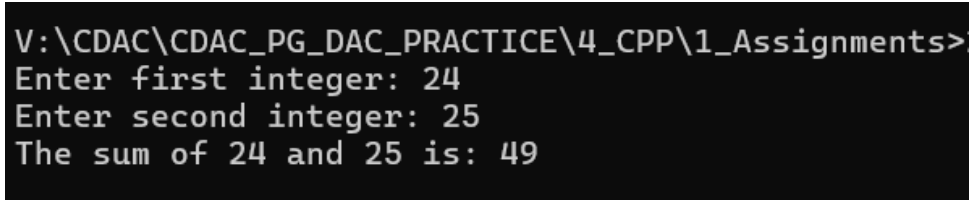
**Task:** Write a C++ program that takes two integer inputs from the user and prints their sum.

**Code:** —

```
1 #include <iostream>
2 using namespace std;
3
4 int main() {
5     int num1, num2, sum;
```

```
6
7     cout << "Enter first integer: ";
8     cin >> num1;
9
10    cout << "Enter second integer: ";
11    cin >> num2;
12
13    sum = num1 + num2;
14
15    cout << "The sum of " << num1 << " and " << num2 << " is: "
16          << sum << endl;
17
18    return 0;
19 }
```

Output: —



```
V:\CDAC\CDAC_PG_DAC_PRACTICE\4_CPP\1_Assignments>
Enter first integer: 24
Enter second integer: 25
The sum of 24 and 25 is: 49
```

### Problem 3:

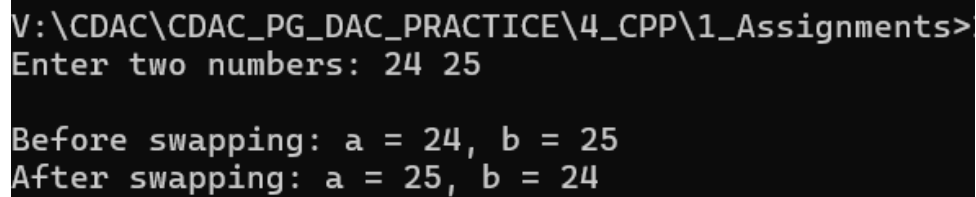
**Task:** Write a C++ program to swap two numbers without using a third variable

Code: —

```
1 #include <iostream>
2 using namespace std;
3
4 int main() {
5     int a, b;
6
7     cout << "Enter two numbers: ";
8     cin >> a >> b;
9
10    cout << "\nBefore swapping: a = " << a << ", b = " << b <<
11          endl;
12
13    // Method 1: Using arithmetic operations
14    a = a + b;
15    b = a - b;
16    a = a - b;
17
18    // Alternative Method 2: Using bitwise XOR (uncomment to test
19    // a = a ^ b;
20    // b = a ^ b;
```

```
20 // a = a ^ b;
21
22 cout << "After swapping: a = " << a << ", b = " << b << endl;
23
24 return 0;
25 }
```

Output: —



```
V:\CDAC\CDAC_PG_DAC_PRACTICE\4_CPP\1_Assignments>
Enter two numbers: 24 25

Before swapping: a = 24, b = 25
After swapping: a = 25, b = 24
```

### Problem 4:

**Task:** Write a C++ program that checks whether a number entered by the user is even or odd.

Code: —

```
1 #include <iostream>
2 using namespace std;
3
4 bool is_even(int n) {
5     return n % 2 == 0;
6 }
7
8 int main() {
9     int n;
10    cout << "Enter a number: ";
11    cin >> n;
12
13    if (is_even(n)) {
14        cout << n << " is even." << endl;
15    } else {
16        cout << n << " is odd." << endl;
17    }
18
19    return 0;
20 }
```

Output: —

```
V:\CDAC\CDAC_PG_DAC_PRACTICE\4_CPP\1_Assignments>
Enter a number: 24
24 is even.

V:\CDAC\CDAC_PG_DAC_PRACTICE\4_CPP\1_Assignments>
Enter a number: 25
25 is odd.
```

## Problem 5:

**Task:** Write a C++ program that takes two numbers and an operator (+, -, \*, /) as input and performs the corresponding operation.

**Code:** —

```
1 #include <iostream>
2 using namespace std;
3
4 int main() {
5     double num1, num2;
6     char op;
7
8     cout << "Enter first number: ";
9     cin >> num1;
10
11     cout << "Enter an operator (+, -, *, /): ";
12     cin >> op;
13
14     cout << "Enter second number: ";
15     cin >> num2;
16
17     double result;
18
19     switch (op) {
20         case '+':
21             result = num1 + num2;
22             cout << "Result: " << num1 << " + " << num2 << " = "
23                 << result << endl;
24             break;
25         case '-':
26             result = num1 - num2;
27             cout << "Result: " << num1 << " - " << num2 << " = "
28                 << result << endl;
29             break;
30         case '*':
31             result = num1 * num2;
32             cout << "Result: " << num1 << " * " << num2 << " = "
33                 << result << endl;
34             break;
35         case '/':
36             if (num2 != 0) {
```

```
34         result = num1 / num2;
35         cout << "Result: " << num1 << " / " << num2 << "
           = " << result << endl;
36     } else {
37         cout << "Error: Division by zero is not allowed!"
           << endl;
38     }
39     break;
40 default:
41     cout << "Invalid operator! Please use +, -, *, or /."
           << endl;
42     break;
43 }
44
45 return 0;
46 }
```

Output: —

```
V:\CDAC\CDAC_PG_DAC_PRACTICE\4_CPP\1_Assignments>
Enter first number: 24
Enter an operator (+, -, *, /): *
Enter second number: 25
Result: 24 * 25 = 600
```

## Problem 6:

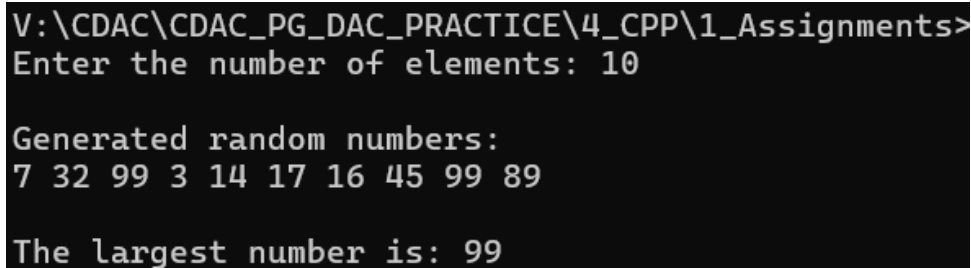
**Task:** Write a C++ program that takes n numbers as input, stores them in an array, and finds the largest number.

Code: —

```
1 #include <iostream>
2 #include <cstdlib> // For rand() and srand()
3 #include <ctime>   // For time()
4 using namespace std;
5
6 int main() {
7     int n;
8
9     cout << "Enter the number of elements: ";
10    cin >> n;
11
12    if (n <= 0) {
13        cout << "Invalid input! Number of elements must be
           greater than zero." << endl;
14        return 1;
15    }
16
17    int arr[n];
```

```
18
19 // Seed the random number generator
20 srand(time(0));
21
22 cout << "\nGenerated random numbers:\n";
23 for (int i = 0; i < n; i++) {
24     arr[i] = rand() % 100; // random values between 0 and 99
25     cout << arr[i] << " ";
26 }
27 cout << endl;
28
29 int largest = arr[0];
30
31 for (int i = 1; i < n; i++) {
32     if (arr[i] > largest) {
33         largest = arr[i];
34     }
35 }
36
37 cout << "\nThe largest number is: " << largest << endl;
38
39 return 0;
40 }
```

Output: —



```
V:\CDAC\CDAC_PG_DAC_PRACTICE\4_CPP\1_Assignments>
Enter the number of elements: 10

Generated random numbers:
7 32 99 3 14 17 16 45 99 89

The largest number is: 99
```

## Problem 7:

**Task:** Write a C++ program that takes an integer input and calculates the sum of its digits.

Code: —

```
1 #include <iostream>
2 using namespace std;
3
4 int main() {
5     int n;
6     cout << "Enter a Number: ";
7     cin >> n;
8     int sum = 0;
9     while (n > 0) {
```

```
10     sum += n % 10;
11     n /= 10;
12 }
13 cout << "Sum of digits: " << sum << endl;
14 return 0;
15 }
```

Output: —

```
V:\CDAC\CDAC_PG_DAC_PRACTICE\4_CPP\1_Assignments>7_sum_of_digits
Enter a Number: 99
Sum of digits: 18
```

## Problem 8:

**Task:** Write a C++ program to take n elements in an array and print them in reverse order.

Code: —

```
1  #include <iostream>
2  using namespace std;
3
4  /**
5   * Reverses the elements of an array in place.
6   *
7   * @param arr The array to reverse.
8   * @param n The number of elements in the array.
9   */
10 void reverse_array(int arr[], int n) {
11     int start = 0;
12     int end = n-1;
13     while (start < end) {
14         int temp = arr[start];
15         arr[start] = arr[end];
16         arr[end] = temp;
17         start++;
18         end--;
19     }
20 }
21
22
23 /**
24  * Example program that demonstrates the use of the reverse_array
25  * function.
26  */
27 int main() {
28     int arr[] = {1, 2, 3, 4, 5};
29     int n = sizeof(arr) / sizeof(arr[0]);
30
31     // Reverse the array in place
```

```
31     reverse_array(arr, n);
32
33     // Print the reversed array
34     for (int i = 0; i < n; i++) {
35         cout << arr[i] << " ";
36     }
37     cout << endl;
38
39     return 0;
40 }
```

Output: —

```
V:\CDAC\CDAC_PG_DAC_PRACTICE\4_CPP\1_Assignments>
5 4 3 2 1
```

## Problem 9:

**Task:** Write a C++ program to check if a given number is palindromic (reads the same forward and backward).

Code: —

```
1 #include <iostream>
2 using namespace std;
3
4 /**
5  * Checks if a given number is a palindrome.
6  *
7  * @param n The number to check.
8  * @return true if the number is a palindrome, false otherwise.
9  */
10 bool is_palindrome_number(int n) {
11     int reversed = 0;
12     int original = n;
13
14     while (n > 0) {
15         int digit = n % 10;
16         reversed = reversed * 10 + digit;
17         n /= 10;
18     }
19
20     return original == reversed;
21 }
22
23 /**
24  * Example program that demonstrates the use of the
25  * is_palindrome_number function.
26  */
27 int main() {
```



```
27     int n;
28     cout << "Enter a number: ";
29     cin >> n;
30
31     if (is_palindrome_number(n)) {
32         cout << n << " is a palindrome number." << endl;
33     } else {
34         cout << n << " is not a palindrome number." << endl;
35     }
36
37     return 0;
38 }
39
40 /**
41  * Checks if a given number is a palindrome.
42  *
43  * A palindrome is a number that reads the same backwards as
44  * forwards.
45  *
46  * @param n The number to check.
47  * @return true if the number is a palindrome, false otherwise.
48  */
```

Output: —

```
V:\CDAC\CDAC_PG_DAC_PRACTICE\4_CPP\1_Assignments>
Enter a number: 143
143 is not a palindrome number.

V:\CDAC\CDAC_PG_DAC_PRACTICE\4_CPP\1_Assignments>
Enter a number: 1771
1771 is a palindrome number.
```

## Problem 10:

**Task:** Write a C++ program to print the Fibonacci series up to n terms.

**Code:** —

```
1 #include <iostream>
2 using namespace std;
3 /**
4  * Prints the first n terms of the Fibonacci series.
5  *
6  * The Fibonacci series is a sequence of numbers in which each
7  * number is the sum of the two preceding numbers.
8  *
9  * @param n The number of terms to print.
10 */
11 void print_fibonacci_series(int n) {
```

```
11     int a = 0, b = 1, c;  
12     if (n == 0) return;  
13     cout << a << " ";  
14     if (n == 1) return;  
15     cout << b << " ";  
16     for (int i = 2; i < n; i++) {  
17         c = a + b;  
18         cout << c << " ";  
19         a = b;  
20         b = c;  
21     }  
22 }  
23  
24 /**  
25  * The main function of the program.  
26  *  
27  * Prompts the user to enter a number, then uses that number to  
28  * print the first n terms of the Fibonacci series.  
29  *  
30  * @return 0 if the program runs successfully.  
31  */  
32 int main() {  
33     int n;  
34     cout << "Enter the number of terms: ";  
35     cin >> n;  
36  
37     // Print the first n terms of the Fibonacci series  
38     print_fibonacci_series(n);  
39  
40     return 0;  
41 }
```

Output: —

```
V:\CDAC\CDAC_PG_DAC_PRACTICE\4_CPP\1_Assignments>  
Enter the number of terms: 10  
0 1 1 2 3 5 8 13 21 34
```

## Problem 11:

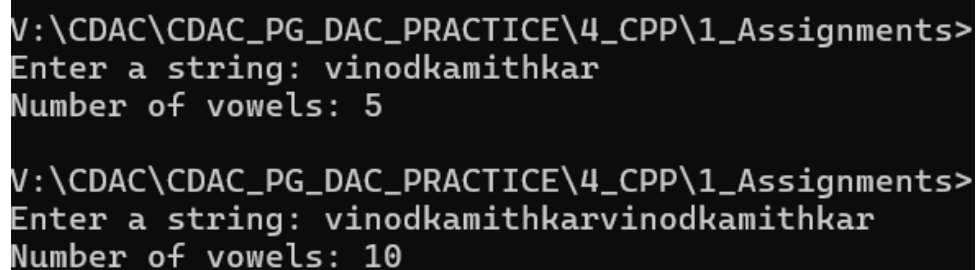
**Task:** Write a C++ program that takes a string as input and counts the number of vowels (a, e, i, o, u).

Code: —

```
1 #include <string>  
2 #include <iostream>  
3 using namespace std;  
4 /**  
5  * Counts the number of vowels in a given string.
```

```
6  *
7  * @param str The string to count the vowels from.
8  * @return The number of vowels in the string.
9  */
10 int count_vowels(const string& str) {
11     int count = 0;
12     for (char c : str) {
13         switch (c) {
14             case 'a':
15             case 'e':
16             case 'i':
17             case 'o':
18             case 'u':
19                 count++;
20                 break;
21             default:
22                 break;
23         }
24     }
25     return count;
26 }
27
28 /**
29  * The main function of the program.
30  *
31  * Prompts the user to enter a string, then uses the count_vowels
32  * function to count the number of vowels in the string.
33  *
34  * @return 0 if the program runs successfully.
35  */
36 int main() {
37     string str;
38     cout << "Enter a string: ";
39     cin >> str;
40     int count = count_vowels(str);
41     cout << "Number of vowels: " << count << endl;
42     return 0;
43 }
```

Output: —



```
V:\CDAC\CDAC_PG_DAC_PRACTICE\4_CPP\1_Assignments>
Enter a string: vinodkamithkar
Number of vowels: 5

V:\CDAC\CDAC_PG_DAC_PRACTICE\4_CPP\1_Assignments>
Enter a string: vinodkamithkarvinodkamithkar
Number of vowels: 10
```

## Problem 12:

**Task:** Write a C++ program to find the GCD of two numbers

**Code:** —

```
1 #include <iostream>
2 using namespace std;
3 /**
4  * Calculates the greatest common divisor (GCD) of two numbers.
5  *
6  * The GCD of two numbers is the largest number that divides both
7  *   of them without leaving a remainder.
8  *
9  * This function uses the Euclidean algorithm to calculate the
10 *   GCD.
11 *
12 * @param a The first number.
13 * @param b The second number.
14 * @return The greatest common divisor of a and b.
15 */
16 int gcd(int a, int b) {
17     // Base case: if b is 0, the GCD is a
18     if (b == 0) return a;
19
20     // Recursive case: calculate the GCD of b and a % b
21     return gcd(b, a % b);
22 }
23
24 /**
25 * The main function of the program.
26 *
27 * Reads two numbers from the user and prints the greatest common
28 *   divisor of the two numbers.
29 *
30 * @return 0 if the program runs successfully.
31 */
32 int main() {
33     int a, b;
34     // Read two numbers from the user
35     cin >> a >> b;
36
37     // Print the greatest common divisor of a and b
38     cout << gcd(a, b) << endl;
39
40     return 0;
41 }
```

**Output:** —

```
V:\CDAC\CDAC_PG_DAC_PRACTICE\4_CPP\1_Assignments>1
24 25
1

V:\CDAC\CDAC_PG_DAC_PRACTICE\4_CPP\1_Assignments>1
18 27
9
```

### Problem 13:

**Task:** Write a C++ program to multiply two matrices.

**Code:** —

```
1 #include <iostream>
2 #include <cstdlib>    // for rand() and srand()
3 #include <ctime>      // for time()
4 using namespace std;
5
6 /**
7  * Multiplies two matrices.
8  *
9  * @param m1 The first matrix.
10 * @param m2 The second matrix.
11 * @return The product of the two matrices.
12 */
13 int** multiply_matrices(int** m1, int rows1, int cols1, int** m2,
14     int rows2, int cols2) {
15     if (cols1 != rows2) {
16         throw invalid_argument("The number of columns in the
17             first matrix must match the number of rows in the
18             second matrix.");
19     }
20
21     // Allocate memory for result matrix
22     int** result = new int*[rows1];
23     for (int i = 0; i < rows1; i++) {
24         result[i] = new int[cols2];
25         for (int j = 0; j < cols2; j++) {
26             result[i][j] = 0; // initialize to 0
27         }
28     }
29
30     // Matrix multiplication
31     for (int i = 0; i < rows1; i++) {
32         for (int j = 0; j < cols2; j++) {
33             for (int k = 0; k < cols1; k++) {
34                 result[i][j] += m1[i][k] * m2[k][j];
35             }
36         }
37     }
38 }
```

```
35
36     return result;
37 }
38
39 // Utility function to print a matrix
40 void print_matrix(int** matrix, int rows, int cols, const string&
    name) {
41     cout << "\n" << name << " (" << rows << "x" << cols << "):\n"
        ";
42     for (int i = 0; i < rows; i++) {
43         for (int j = 0; j < cols; j++) {
44             cout << matrix[i][j] << "\t";
45         }
46         cout << endl;
47     }
48 }
49
50 int main() {
51     srand(time(0)); // Seed random number generator
52
53     int rows1, cols1, rows2, cols2;
54
55     cout << "Enter number of rows and columns for the first
        matrix: ";
56     cin >> rows1 >> cols1;
57
58     cout << "Enter number of rows and columns for the second
        matrix: ";
59     cin >> rows2 >> cols2;
60
61     if (cols1 != rows2) {
62         cout << "Matrix multiplication not possible! Columns of
            first must equal rows of second.\n";
63         return 1;
64     }
65
66     // Allocate and fill first matrix with random values
67     int** m1 = new int*[rows1];
68     for (int i = 0; i < rows1; i++) {
69         m1[i] = new int[cols1];
70         for (int j = 0; j < cols1; j++) {
71             m1[i][j] = rand() % 10; // random values between 0
                9
72         }
73     }
74
75     // Allocate and fill second matrix with random values
76     int** m2 = new int*[rows2];
77     for (int i = 0; i < rows2; i++) {
78         m2[i] = new int[cols2];
79         for (int j = 0; j < cols2; j++) {
```

```

80         m2[i][j] = rand() % 10;
81     }
82 }
83
84 // Display input matrices
85 print_matrix(m1, rows1, cols1, "Matrix 1");
86 print_matrix(m2, rows2, cols2, "Matrix 2");
87
88 // Multiply matrices
89 int** result = multiply_matrices(m1, rows1, cols1, m2, rows2,
90                                 cols2);
91
92 // Display result
93 print_matrix(result, rows1, cols2, "Resultant Matrix (M1 x M2
94                )");
95
96 // Free allocated memory
97 for (int i = 0; i < rows1; i++) delete[] m1[i];
98 delete[] m1;
99
100 for (int i = 0; i < rows2; i++) delete[] m2[i];
101 delete[] m2;
102
103 for (int i = 0; i < rows1; i++) delete[] result[i];
104 delete[] result;
105
106 return 0;
107 }

```

Output: —

```

V:\CDAC\CDAC_PG_DAC_PRACTICE\4_CPP\1_Assignments>13_matrices_multiplication
Enter number of rows and columns for the first matrix: 2 2
Enter number of rows and columns for the second matrix: 2 2

Matrix 1 (2x2):
9      9
9      2

Matrix 2 (2x2):
1      1
8      8

Resultant Matrix (M1 x M2) (2x2):
81     81
25     25

V:\CDAC\CDAC_PG_DAC_PRACTICE\4_CPP\1_Assignments>13_matrices_multiplication
Enter number of rows and columns for the first matrix: 2 3
Enter number of rows and columns for the second matrix: 9 7
Matrix multiplication not possible! Columns of first must equal rows of second.

```

## Problem 14:

**Task:** A number is an Armstrong number if the sum of its digits raised to the power of the number of digits is equal to the number itself (e.g.,  $153 = 1^3 + 5^3 + 3^3$ ). Write a C++ program to check if a number is Armstrong.

Code: —

```
1 #include <iostream>
2 #include <cmath>
3 using namespace std;
4
5 bool is_armstrong_number(int n) {
6     int digits = 0;
7     int original = n;
8     int sum = 0;
9
10    while (n > 0) {
11        n /= 10;
12        digits++;
13    }
14
15    n = original;
16    while (n > 0) {
17        int digit = n % 10;
18        sum += pow(digit, digits);
19        n /= 10;
20    }
21
22    return sum == original;
23 }
24
25 int main() {
26     int n;
27     cout << "Enter a number: ";
28     cin >> n;
29
30     if (is_armstrong_number(n)) {
31         cout << n << " is an Armstrong number." << endl;
32     } else {
33         cout << n << " is not an Armstrong number." << endl;
34     }
35
36     return 0;
37 }
```

Output: —

```
V:\CDAC\CDAC_PG_DAC_PRACTICE\4_CPP\1_Assignments>
Enter a number: 125
125 is not an Armstrong number.

V:\CDAC\CDAC_PG_DAC_PRACTICE\4_CPP\1_Assignments>
Enter a number: 153
153 is an Armstrong number.
```



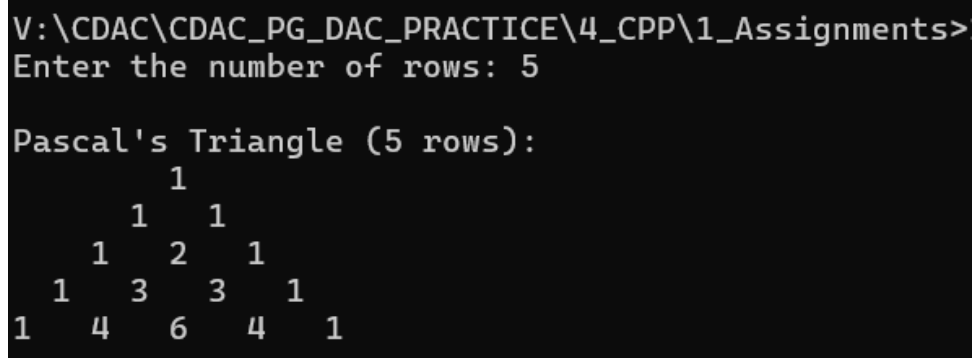
**Problem 15:**

**Task:** Write a C++ program to print Pascal's triangle up to n rows.

**Code:** —

```
1 #include <iostream>
2 using namespace std;
3
4 // Function to print Pascal's Triangle
5 void print_pascal_triangle(int n) {
6     for (int i = 0; i < n; i++) {
7         int value = 1; // First element of every row is 1
8
9         // Print leading spaces for alignment
10        for (int space = 0; space < n - i - 1; space++) {
11            cout << " ";
12        }
13
14        // Print values in the row
15        for (int j = 0; j <= i; j++) {
16            cout << value << " ";
17            value = value * (i - j) / (j + 1); // Compute next
18            value in row
19        }
20        cout << endl;
21    }
22 }
23
24 int main() {
25     int n;
26     cout << "Enter the number of rows: ";
27     cin >> n;
28     cout << "\nPascal's Triangle (" << n << " rows):\n";
29     print_pascal_triangle(n);
30
31     return 0;
32 }
```

**Output:** —



```
V:\CDAC\CDAC_PG_DAC_PRACTICE\4_CPP\1_Assignments>
Enter the number of rows: 5

Pascal's Triangle (5 rows):
      1
     1 1
    1 2 1
   1 3 3 1
  1 4 6 4 1
```