THE UNIVERSITY OF AZAD JAMMU AND KASHMIR, MUZAFFARABAD

Department of Software Engineering

Submitted to:	Engr. Sidra Rafique
Course Title:	Data Structure and Algorithm
Course Code:	CS-2101
Session:	2024-2028
Lab No:	01
Roll No:	2024-SE-15
Submitted By:	Shahzad Ahmed Awan
Submission date:	October 25, 2025

Table of Contents

Table of Figures

Lab 01 – Array Operations: Sum and Maximum Calculation

---

1. Objective

The objective of this laboratory exercise is to develop a C++ program that accepts five integer inputs, stores them in an array, computes the cumulative sum, and determines the largest value among them.
This task aims to strengthen understanding of basic data structures, iterative control structures, and conditional operations in C++.

---

2. Background

Arrays are one of the most fundamental linear data structures used to store and process multiple values of the same type.
They provide indexed access to data elements, enabling efficient traversal and manipulation through iterative constructs such as loops.

This lab demonstrates how arrays can be combined with control structures to perform essential operations such as summation and maximum-value detection. It reinforces the logic of sequential processing and enhances the understanding of how computational problems can be solved through structured programming techniques.

---

3. Algorithm

1. Start the program.
2. Declare an integer array numbers[5] to hold five values.
3. Initialize variable sum to 0.
4. Declare variable max to store the largest number.
5. Prompt the user to input five integers.
6. Store each value in the array using a loop.
7. Assign the first element of the array to max.
8. Traverse the array again to:
    - Add each element to sum.
    - Compare each element with max; update if a greater value is found.
9. Display the entered numbers, the calculated sum, and the maximum value.
10. End the program.

4. Source Code

The following C++ program was written and executed as part of this LAB.
It implements the algorithm described above to read five numbers, calculate their sum, and find the maximum value

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

int main() {
    int numbers[5];  // Declare an array of 5 integers
    int sum = 0, max;

    cout << "Enter 5 numbers: " << endl;

    // Input values into the array
    for (int i = 0; i < 5; i++) {
        cout << "Number " << i + 1 << ": ";
        cin >> numbers[i];
    }

    // Initialize max with the first element
    max = numbers[0];

    // Calculate sum and find maximum
    for (int i = 0; i < 5; i++) {
        sum += numbers[i];
        if (numbers[i] > max)
            max = numbers[i];
    }

    // Display results
    cout << "\nYou entered: ";
    for (int i = 0; i < 5; i++) {
        cout << numbers[i] << " ";
    }

    cout << "\nSum of numbers = " << sum;
    cout << "\nMaximum number = " << max << endl;

    return 0;
}
```

Figure 1: Program Source Code

5. Explanation of Code

5.1 Header and Namespace

The directive #include <iostream> includes the standard input/output stream library, allowing the use of cin and cout for data entry and display.
The statement using namespace std; enables direct access to standard library objects without prefixing them with std::

5.2 Variable Declaration

An integer array numbers[5] is declared to store five numeric inputs.
The variable sum is initialized to 0 to accumulate the total, while max is declared to hold the highest value in the array.

This approach ensures that memory is allocated for all five integers, enabling efficient indexing and retrieval during processing.

5.3 Input Section

A for loop is used to read five integer values from the user.
Each iteration prompts the user to enter a number, which is then stored in the array at the corresponding index position.

Figure 2: Input from user Section

5.4 Initialization of Maximum

Before comparison begins, the first element of the array (numbers[0]) is assigned to max.
This step provides a valid reference point for subsequent comparisons during traversal.

5.5 Processing Logic

A second loop iterates through all elements to compute the total sum and determine the maximum value.
For every element, its value is added to sum, and if it exceeds the current max, the max variable is updated accordingly.
This ensures that both computations are performed in a single pass, maintaining linear time complexity *O(n)*.

5.6 Display Section

After computation, the program outputs the entered numbers, followed by the calculated sum and maximum value.
This section ensures that users can verify their inputs alongside the computed results.

6. Output

Upon execution, the program prompts the user to input five integers.
After all values are entered, it displays:

- The list of numbers entered.
- The computed total sum of all elements.
- The largest number among them.

This structured output verifies that the program logic correctly performs array traversal, aggregation, and comparison.

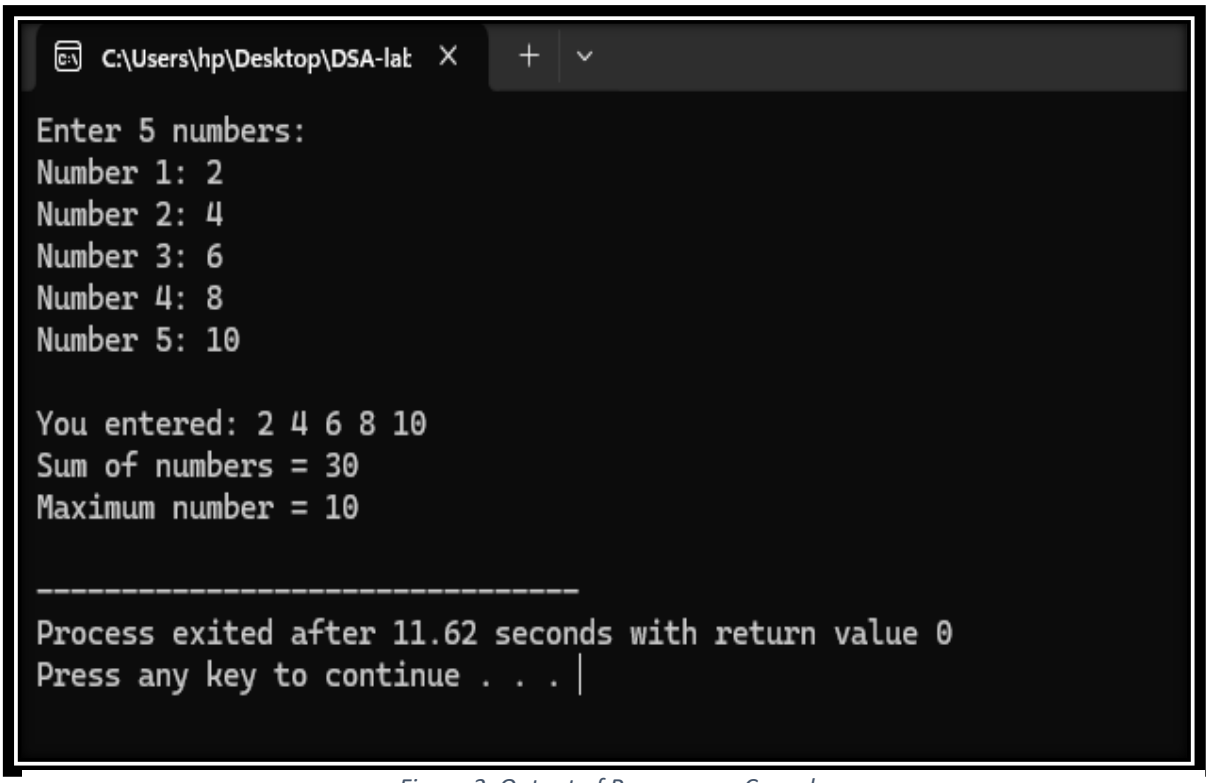

Figure 3: Output of Program on Console

---

7. Conclusion

This lab exercise effectively demonstrated the implementation of array-based operations using iterative and conditional constructs in C++.
The program accurately reads multiple inputs, stores them in an array, calculates their sum, and identifies the largest number through sequential comparison.

The practical implementation reinforced core programming concepts, including array manipulation, decision control, and iteration.
The algorithm used maintains linear time complexity, ensuring both accuracy and efficiency.

Overall, this LAB enhanced the conceptual and practical understanding of how arrays can be used to perform real-world computational tasks with minimal complexity.

---

8. Reflection

This Lab provided valuable hands-on experience and practical reinforcement of theoretical programming knowledge.

Through this activity, I learned and practiced the following:

- How to declare and use arrays effectively in C++.
- How to apply looping structures to process data stored in arrays.
- The use of conditional statements for comparison and decision-making.
- The importance of initializing variables correctly before iteration.
- How linear algorithms ensure efficiency in small data-processing tasks.
- How structured programming principles apply in real implementations.
- The significance of clear and systematic program output for verification.

This Lab not only improved technical skills in array handling but also strengthened logical thinking and debugging ability — essential for advancing toward more complex data structure problems in future labs.