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:	02
Roll No:	2024-SE-15
Submitted By:	Shahzad Ahmed Awan
Submission date:	October 25, 2025

Table of Contents

Lab 02 – Pointer-Based Array Operations: Sum and Maximum Calculation 3

1. Objective 3
2. Background 3
3. Algorithm 3
4. Source Code 4
5. Explanation of Code 4
   - 5.1 Pointer Declaration 4
   - 5.2 Input Using Pointer 4
   - 5.3 Initialization of Maximum 5
   - 5.4 Calculation of Sum and Maximum 5
   - 5.5 Displaying Results 5
6. Output 6
7. Conclusion 7
8. Reflection 7

Table of Figures

Figure 1: Program Source Code 4

Figure 2: Input using the Pointers 5

Figure 3: Calculation of sum and max using Pointers 5

Figure 4: Displaying the Result on the Console 6

Figure 5: Screenshot of the Output on the Console 6

Lab 02 – Pointer-Based Array Operations: Sum and Maximum Calculation

1. Objective

The objective of this **LAB** is to develop a C++ program that performs the same operations as the previous array-based lab — reading five integers, finding their sum, and identifying the maximum number — but this time using **pointers** instead of direct array indexing.
This activity enhances understanding of how pointers can be used to access and manipulate array elements through memory addresses.

2. Background

Pointers are variables that store the memory address of another variable.
In C++, they can be used to access and modify data indirectly, allowing for more flexible manipulation of arrays and variables.

This **LAB** demonstrates how pointer arithmetic can replace array indexing while producing identical results. The program uses a basic pointer-based approach to perform input, computation, and output operations efficiently.

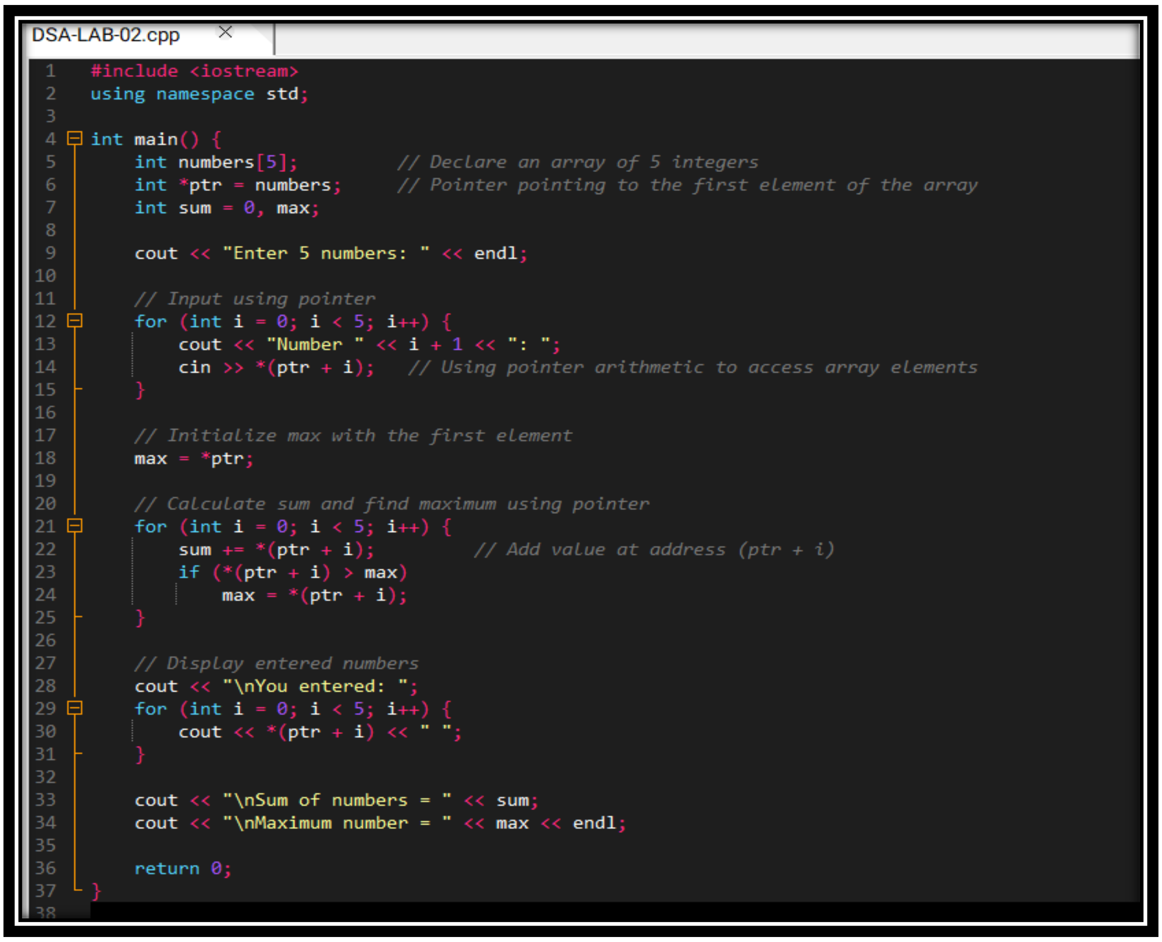
3. Algorithm

1. Start the program.
2. Declare an integer array named numbers[5] to hold five values.
3. Declare a pointer variable ptr and assign it the address of the first element of the array.
4. Initialize a variable sum to zero.
5. Declare a variable max to store the largest number.
6. Use a loop to input five integers using the pointer.
7. Initialize max with the first element accessed through the pointer.
8. Use another loop to:
   - Add each value to sum.
   - Compare each value with max; update if a greater value is found.
9. Display all entered numbers, total sum, and the maximum number using the pointer.
10. End the program.

4. Source Code

The following C++ program was written and executed as part of this **LAB**.
It performs the same operations as the array-based version but uses pointer arithmetic to access and process data.

Below is the complete program code:


DSA-LAB-02.cpp

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

int main() {
    int numbers[5];          // Declare an array of 5 integers
    int *ptr = numbers;      // Pointer pointing to the first element of the array
    int sum = 0, max;

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

    // Input using pointer
    for (int i = 0; i < 5; i++) {
        cout << "Number " << i + 1 << ": ";
        cin >> *(ptr + i);   // Using pointer arithmetic to access array elements
    }

    // Initialize max with the first element
    max = *ptr;

    // Calculate sum and find maximum using pointer
    for (int i = 0; i < 5; i++) {
        sum += *(ptr + i);          // Add value at address (ptr + i)
        if (*(ptr + i) > max)
            max = *(ptr + i);
    }

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

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

    return 0;
}
```

Figure 1: Program Source Code

---

5. Explanation of Code

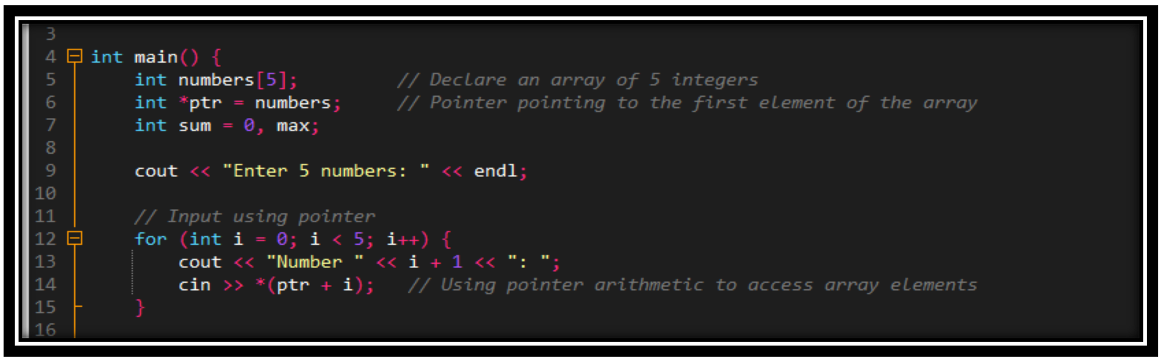
5.1 Pointer Declaration

The statement *int pointer variable ptr = numbers;* declares a pointer named **ptr** and assigns it the address of the first element of the array **numbers**.
In C++, the array name automatically represents the address of its first element, which means **ptr** can be used to access all elements of the array through pointer arithmetic.

5.2 Input Using Pointer

During input, the loop reads user values using pointer arithmetic.
The expression *(ptr + i)* moves the pointer to the i-th position, and by placing an asterisk (*)

before it, the value at that position is accessed.
Thus, the input statement effectively stores each user-entered number in the correct memory location.



```
int main() {
    int numbers[5];         // Declare an array of 5 integers
    int *ptr = numbers;     // Pointer pointing to the first element of the array
    int sum = 0, max;

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

    // Input using pointer
    for (int i = 0; i < 5; i++) {
        cout << "Number " << i + 1 << ": ";
        cin >> *(ptr + i);   // Using pointer arithmetic to access array elements
    }
```

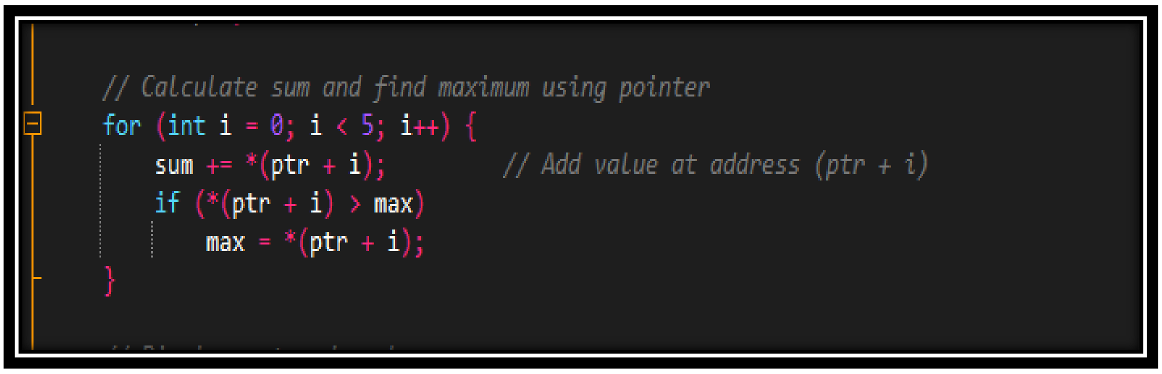
Figure 2: Input using the Pointers

5.3 Initialization of Maximum

The first element of the array is accessed through *ptr* and stored in the variable **max**.
This ensures that the initial maximum value is valid and can be compared against subsequent elements during iteration.

5.4 Calculation of Sum and Maximum

The program uses another loop to perform two operations simultaneously.
In each iteration, the value pointed to by *(ptr + i)* is added to **sum**, and compared with **max**.
If the current value is greater than **max**, the **max** variable is updated.
By the end of the loop, **sum** holds the total of all numbers, and **max** holds the largest value.

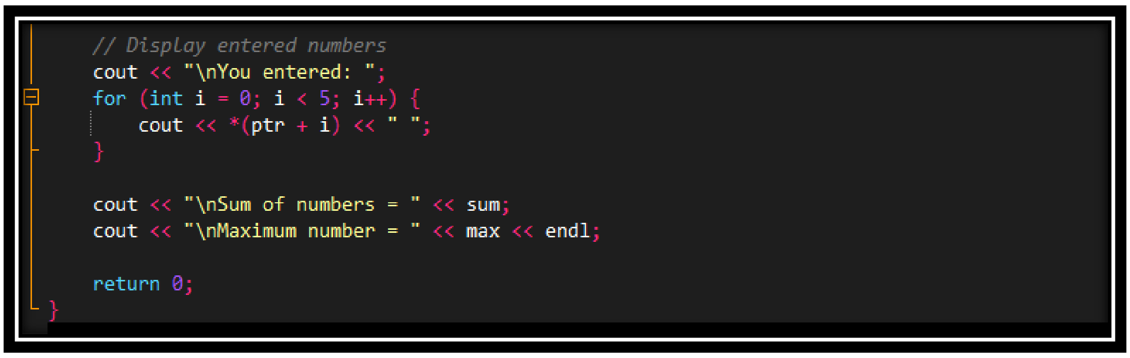


```
// Calculate sum and find maximum using pointer
for (int i = 0; i < 5; i++) {
    sum += *(ptr + i);          // Add value at address (ptr + i)
    if (*(ptr + i) > max)
        max = *(ptr + i);
}
```

Figure 3: Calculation of sum and max using Pointers

5.5 Displaying Results

Finally, the program displays all elements of the array by dereferencing the pointer in each iteration.



```
    // Display entered numbers
    cout << "\nYou entered: ";
    for (int i = 0; i < 5; i++) {
        cout << *(ptr + i) << " ";
    }

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

    return 0;
}
```

Figure 4: Displaying the Result on the Console

This confirms that both input and output can be handled entirely through pointer arithmetic, producing the same output as an array-based approach.

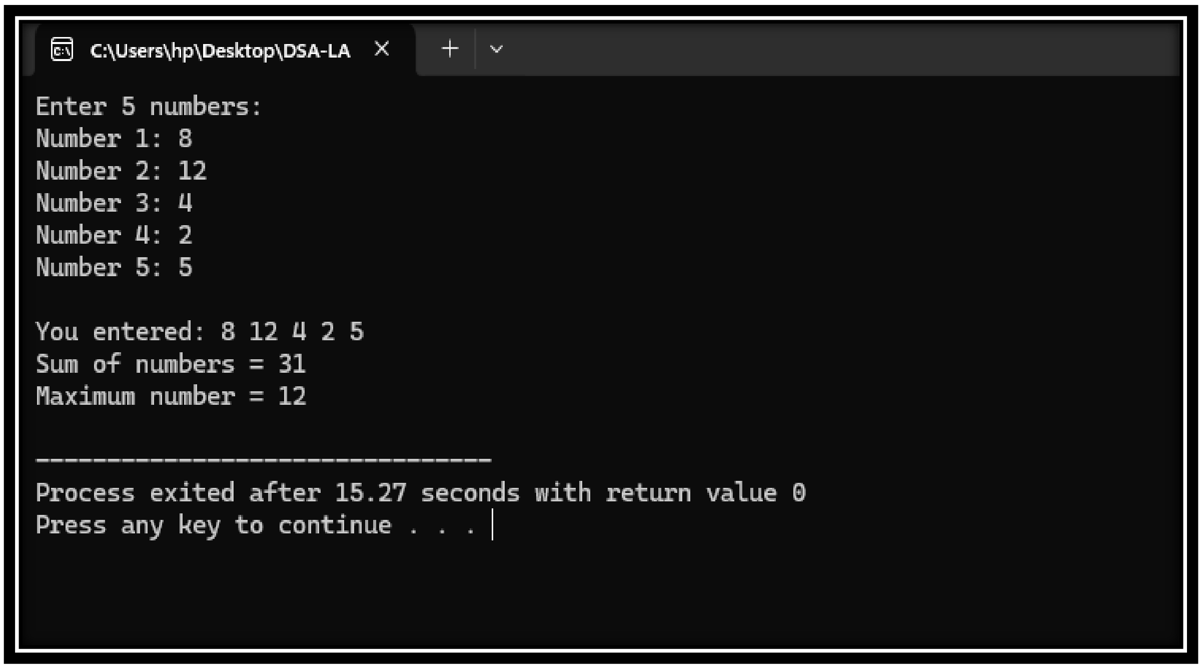
---

6. Output

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

- The numbers entered by the user.
- The total sum of all numbers.
- The largest number among them.

This confirms that the pointer-based approach produces the same results as the traditional array method.



```
Enter 5 numbers:
Number 1: 8
Number 2: 12
Number 3: 4
Number 4: 2
Number 5: 5

You entered: 8 12 4 2 5
Sum of numbers = 31
Maximum number = 12

--------------------------------
Process exited after 15.27 seconds with return value 0
Press any key to continue . . .
```

Figure 5: Screenshot of the Output on the Console

7. Conclusion

This Lab successfully demonstrated how pointers can be used to perform basic array operations such as summation and finding the maximum number.
The pointer-based approach worked equivalently to the array indexing method while providing deeper insight into how data is accessed through memory addresses.

The program execution verified that pointer arithmetic is a reliable and efficient way to navigate through array elements.
Overall, this Lab reinforced basic programming concepts and improved understanding of how pointers interact with arrays in C++.

---

8. Reflection

This Lab provided valuable hands-on experience with pointers in C++.
It helped bridge the conceptual gap between arrays and memory addresses.
From this lab, I learned and practiced the following:

- How a pointer variable can refer to the first element of an array.
- How pointer arithmetic can be used to access array elements.
- How dereferencing works in reading and writing data.
- How to perform summation and comparison operations through pointers.
- The difference between array indexing and pointer notation in practice.
- The importance of clear output presentation for validation.

This Lab improved my confidence in using pointers effectively for data manipulation while reinforcing my understanding of arrays, loops, and conditionals.