جامعة المستقبل
AL MUSTAQBAL UNIVERSITY

قسم الامن السيبراني
DEPARTMENT OF CYBER SECURITY

SUBJECT:

STRUCTURED PROGRAMMING

CLASS:

1ST STAGE

LECTURER:

DR. ABDULKADHEM A. ABDULKADHEM

LECTURE: (8)

Arrays in Functions

1. Introduction to Arrays and Functions

In C++, arrays are used to store multiple values of the same type. Sometimes we want to process arrays inside functions (e.g., to compute the sum, sort, or search for elements).

But unlike simple variables, arrays behave **differently** when passed to functions.

2. How Arrays Are Passed to Functions

When you pass an array to a function:

- **Only the address (pointer to the first element)** is passed.
- The function does **not** get a full copy of the array.

This means:

- Arrays are **passed by reference**.
- Any change made to the array inside the function affects the original array.

3. Syntax: Declaring a Function That Takes an Array

```
// Function declaration
void printArray(int arr[], int size);
```

Or equivalently:

```
void printArray(int* arr, int size);
```

Both declarations mean the same thing. The array parameter is actually treated as a pointer.

4. Example: Printing an Array

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

void printArray(int arr[], int size) {
    for (int i = 0; i < size; i++) {
        cout << arr[i] << " ";
    }
    cout << endl;
}
int main() {
    int data[] = {1, 2, 3, 4, 5};
    printArray(data, 5);
    return 0;
}
```

 Note: You must pass the size separately, since arrays in C++ do not carry their size.

5. Modifying Arrays Inside Functions

Because arrays are passed by reference, we can modify them:

```
void doubleValues(int arr[], int size) {
    for (int i = 0; i < size; i++) {
        arr[i] *= 2;
    }
}

int main() {
    int numbers[] = {1, 2, 3};
    doubleValues(numbers, 3);
    printArray(numbers, 3); // Output: 2 4 6
}
```

6. Multidimensional Arrays

For 2D arrays, you must specify the second dimension in the parameter:

```
void printMatrix(int matrix[][3], int rows) {
    for (int i = 0; i < rows; i++) {
        for (int j = 0; j < 3; j++) {
            cout << matrix[i][j] << " ";
        }
        cout << endl;
    }
}
```

 You must specify column size (`[3]`), but not row size.

7. Common Mistakes and Gotchas

Mistake	Correction
Not passing size of the array	Always pass array size explicitly
Thinking the array is copied	Remember it's passed by reference
Modifying array unintentionally	Be cautious — changes affect original array
Forgetting `[]` or `*` in parameter	Use `int arr[]` or `int* arr`

8. Quiz / Practice Questions

Q1: What is the output of the following code?

```
void update(int arr[], int size) {
    arr[0] = 99;
}
int main() {
    int x[] = {1, 2, 3};
    update(x, 3);
    cout << x[0];
}
```

 Answer: `99` — because arrays are passed by reference.

9. Summary

- Arrays are always passed to function by reference.
- Changes inside functions affect the original array.
- Always pass the size of the array.
- Use `int arr[]` or `int* arr` in function parameters.
- For multidimensional arrays, specify all but the first dimension.

10. Assignment (Homework)

Write a C++ program that:

1. Takes an array of integers from the user.
2. Passes the array to a function that:
    - Computes the sum.
    - Finds the largest element.
    - Reverses the array.