Al-Mustaqbal University

College of Engineering Technology

Cybersecurity Techniques Engineering Department

Programming Essential

Lecture 7

Arrays (One-Dimensional) and Arrays (Two-Dimensional)

PhD. BEng. Ahmed Hasan Janabi
PhD in Cybersecurity
Email: Ahmed.Janabi@uomus.edu.iq

Objectives

By the end of this lecture, students will be able to:

- Identify the concept and purpose of arrays in C++
- Distinguish between one-dimensional and two-dimensional arrays
- Access and modify array elements using indexing
- Apply loops to process array data efficiently
- Implement nested loops to work with two-dimensional arrays
- Write simple C++ programs that use arrays

Think

- Cybersecurity systems don’t store one value — they store many.
- If a system monitors 100 users, do we create 100 variables?

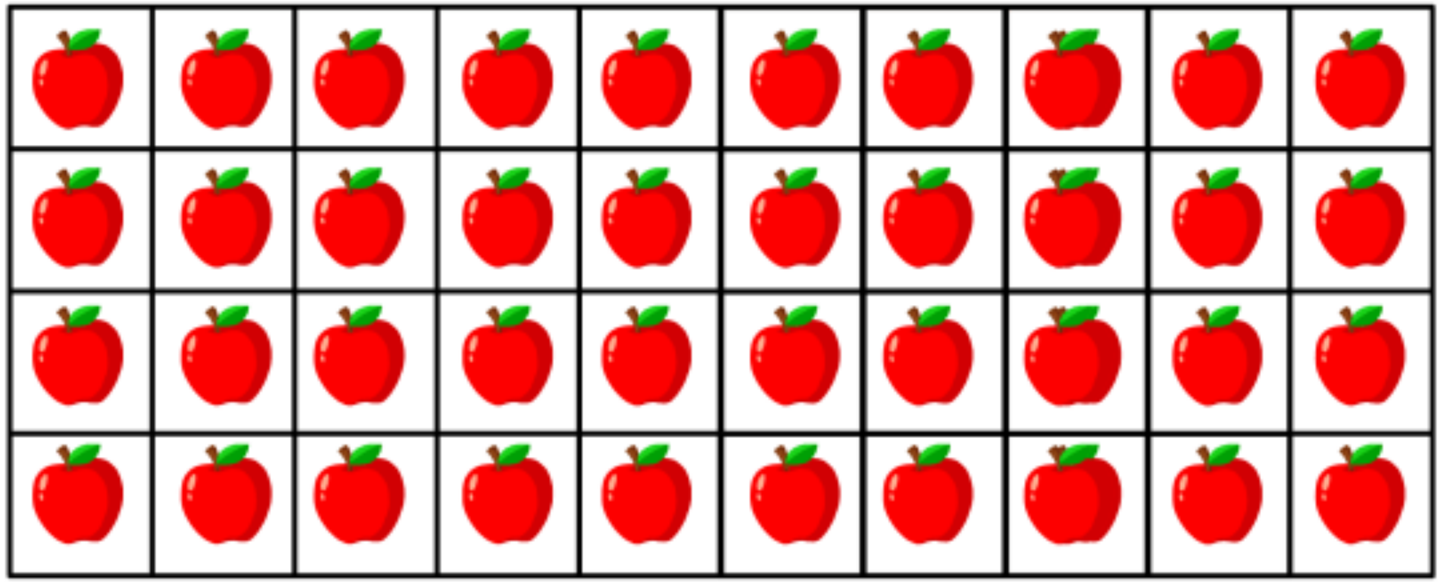
Part 1: What Is an Array?

An **array** is:

- A group of variables of the **same type**, stored under **one name**.

Example:

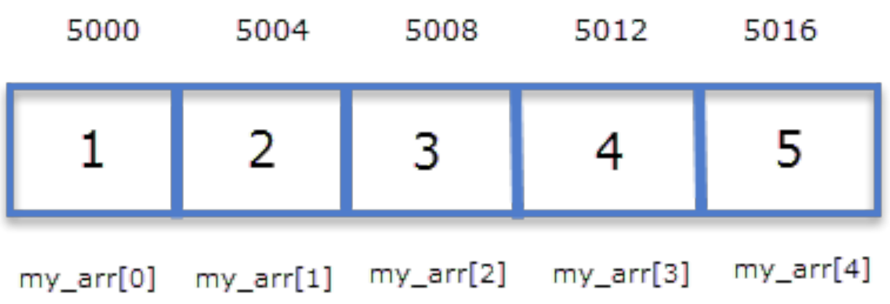
- User IDs
- Login attempts
- Security scores
- Scan results

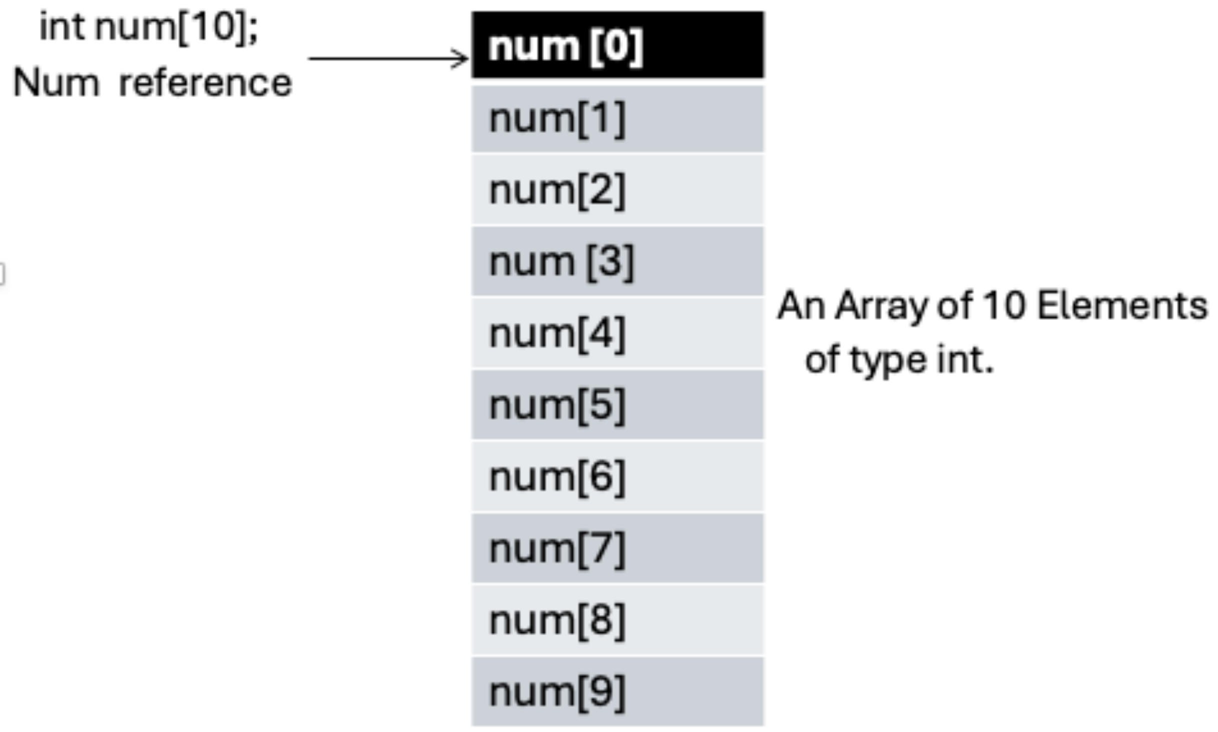


Part 2: One-Dimensional Arrays

A 1D array is like:

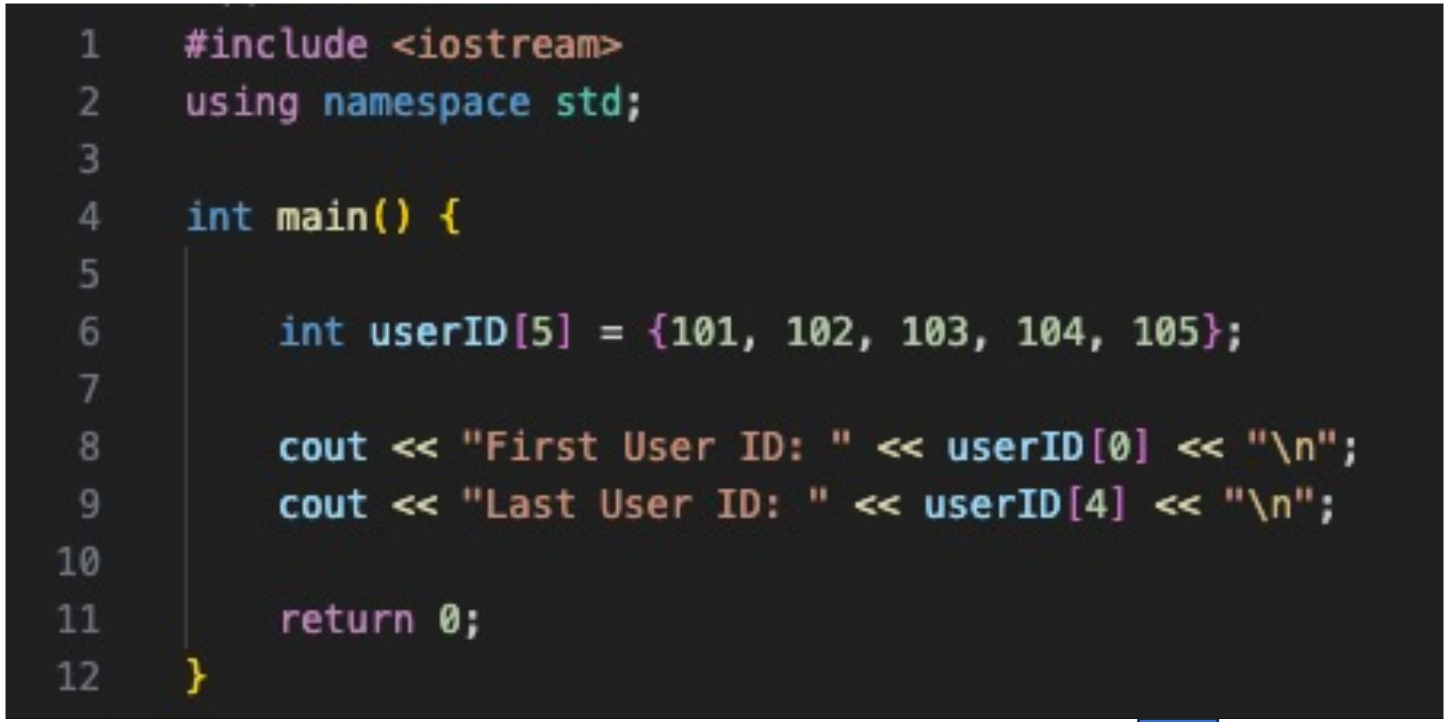
- ✓ One row with multiple elements
- ✓ A list of values
- ✓ A single line of data





Part 2: One-Dimensional Arrays

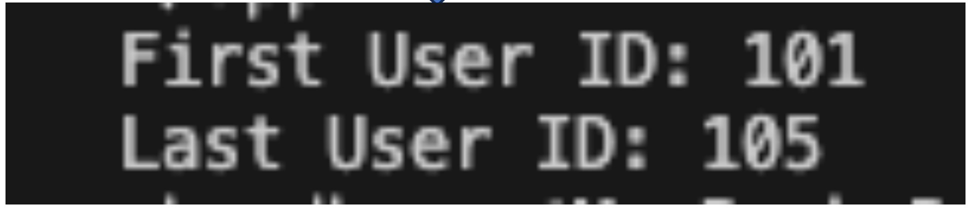
❖ Example: User IDs List



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

int main() {

    int userID[5] = {101, 102, 103, 104, 105};

    cout << "First User ID: " << userID[0] << "\n";
    cout << "Last User ID: " << userID[4] << "\n";

    return 0;
}
```

Indexing is critical in cybersecurity — one mistake can expose wrong data.

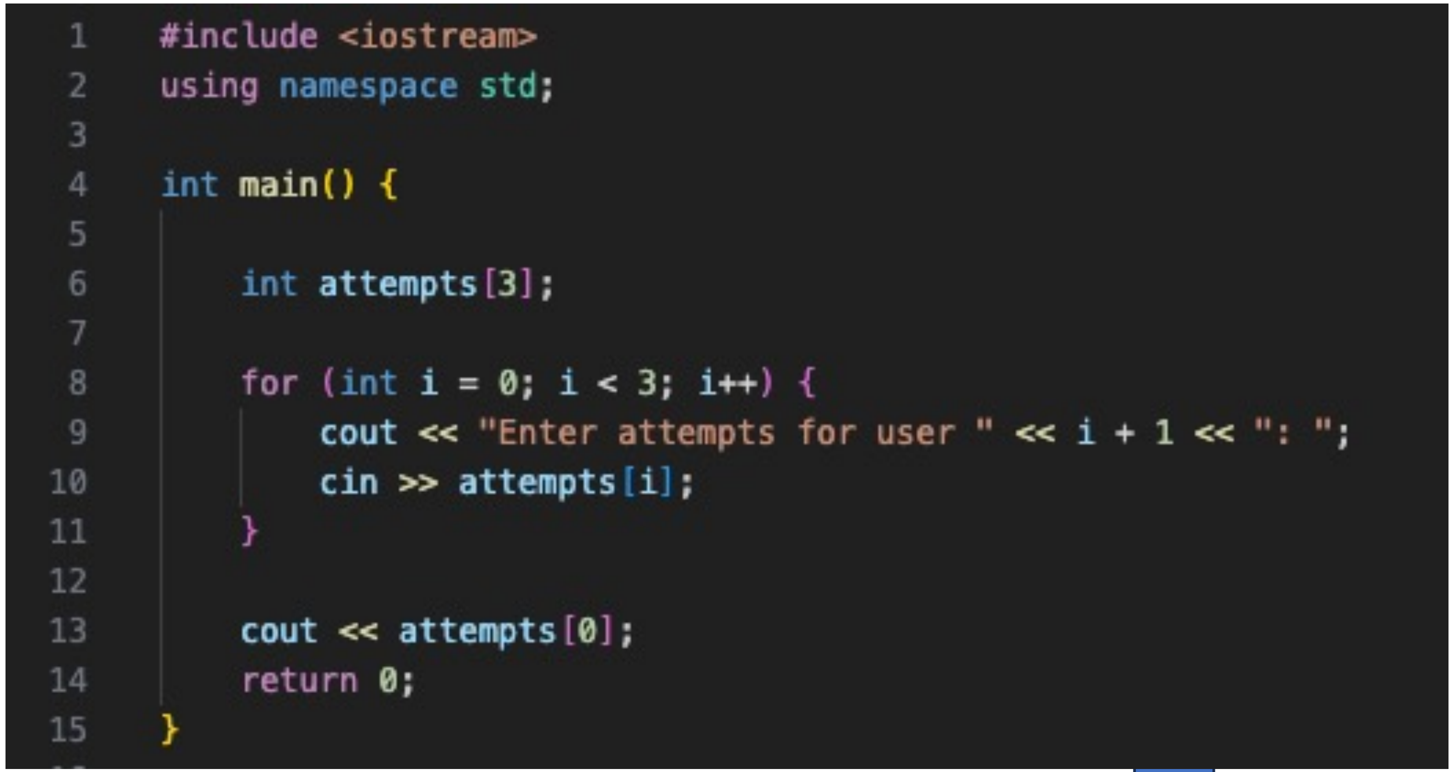
Output



```
First User ID: 101
Last User ID: 105
```

Part 3: Filling an Array Using a Loop

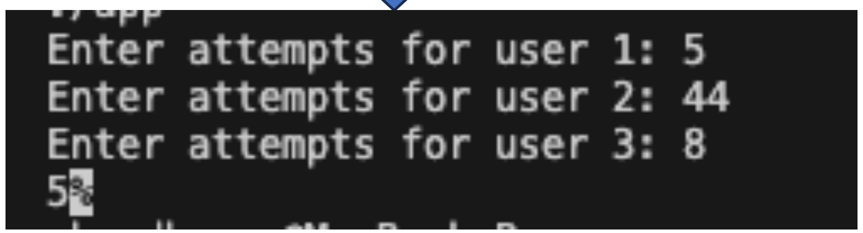
❖ Example: Login Attempts Counter



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

int main() {

    int attempts[3];

    for (int i = 0; i < 3; i++) {
        cout << "Enter attempts for user " << i + 1 << ": ";
        cin >> attempts[i];
    }

    cout << attempts[0];
    return 0;
}
```

Loops + arrays = automation.

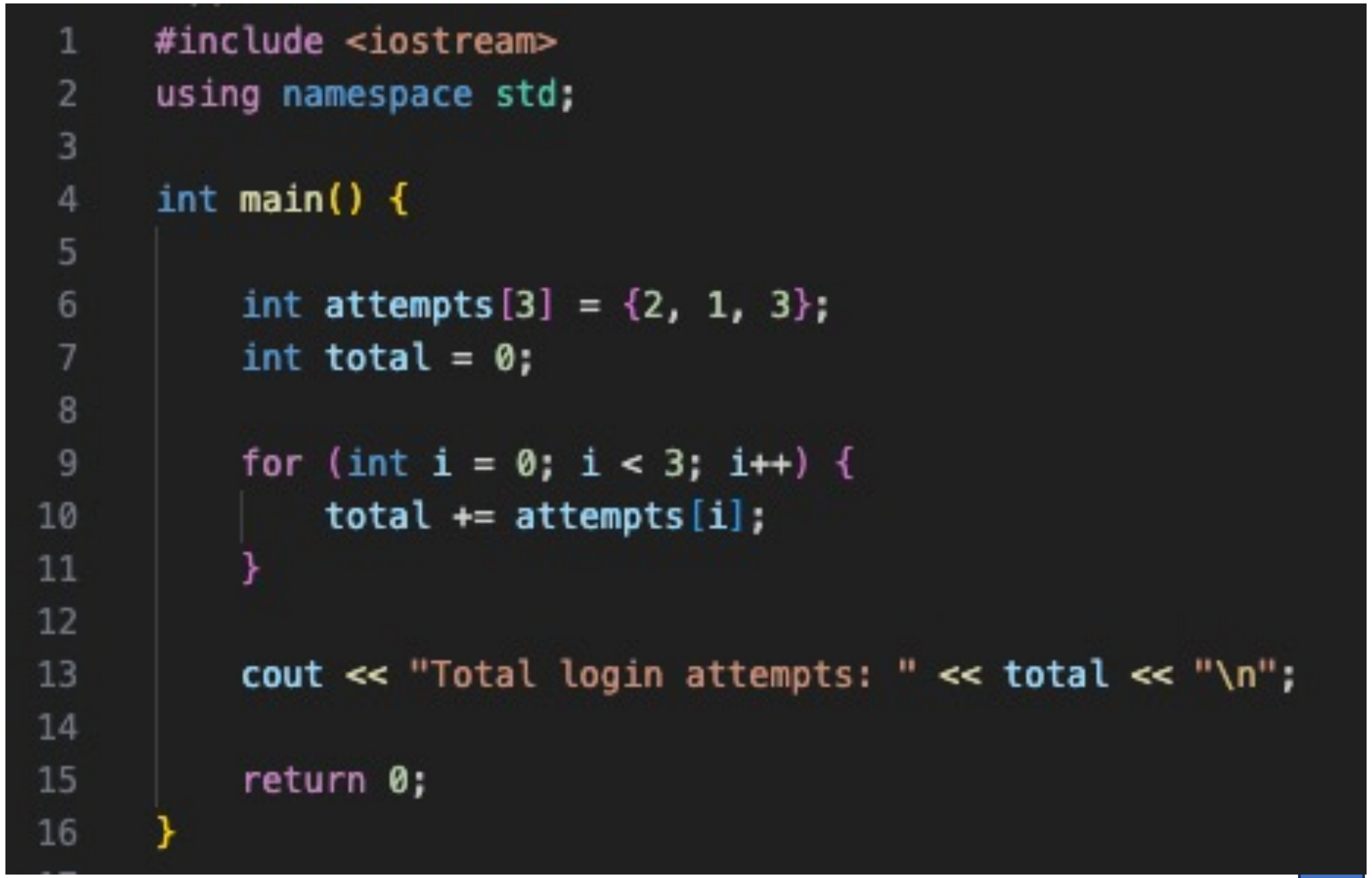
Output



```
./app
Enter attempts for user 1: 5
Enter attempts for user 2: 44
Enter attempts for user 3: 8
5%
```

Part 4: Processing Array Data

❖ Example: Total Login Attempts



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

int main() {

    int attempts[3] = {2, 1, 3};
    int total = 0;

    for (int i = 0; i < 3; i++) {
        total += attempts[i];
    }

    cout << "Total login attempts: " << total << "\n";

    return 0;
}
```

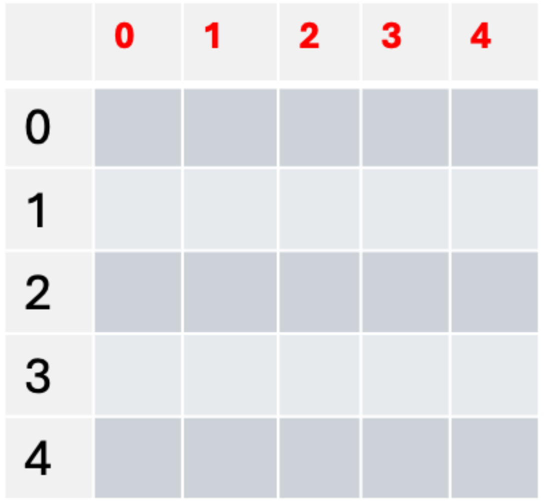
Security systems analyse stored data — this is how.

Output

?

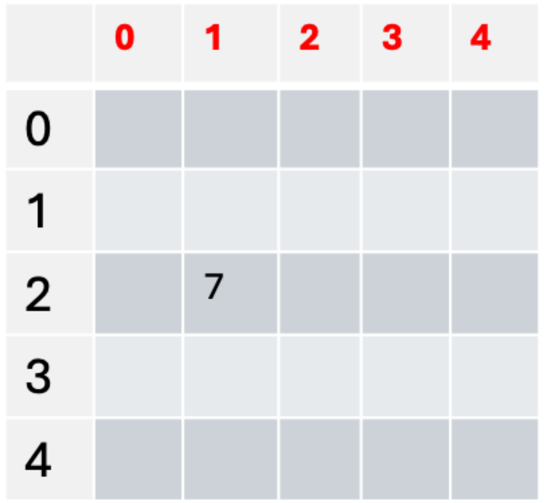
Part 5: Two-Dimensional Arrays

- A **2D array** is like:
  - ✓ A **table**
  - ✓ Rows and columns
- **Used for:**
  - ✓ Network scans
  - ✓ User-permission tables
  - ✓ Logs by day and user



	0	1	2	3	4
0					
1					
2					
3					
4					

```
int matrix[5] [5];
```

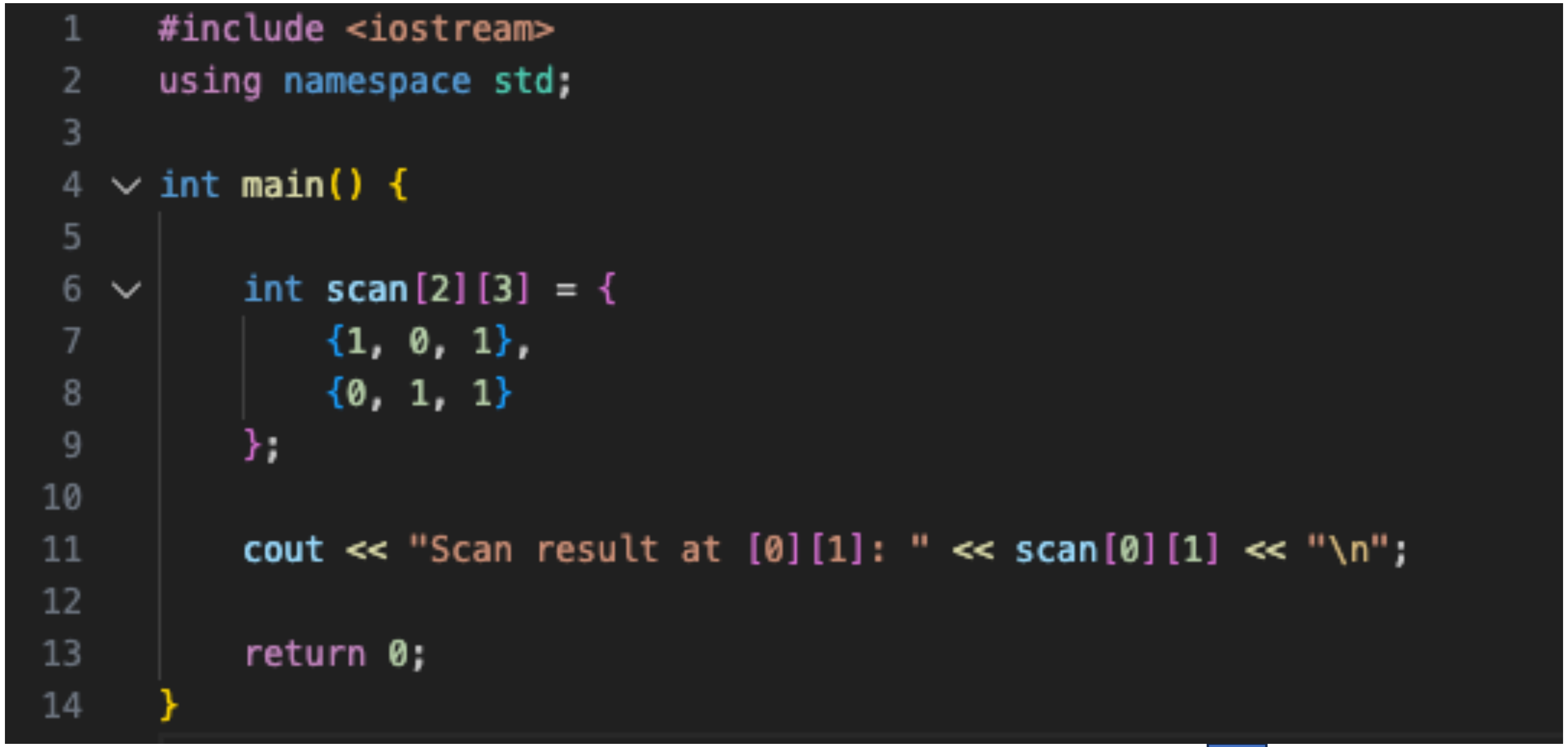


	0	1	2	3	4
0					
1					
2		7			
3					
4					

```
matrix[2] [1] = 7
```

Part 5: Two-Dimensional Arrays

❖ Example: Network Scan Matrix



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

int main() {

    int scan[2][3] = {
        {1, 0, 1},
        {0, 1, 1}
    };

    cout << "Scan result at [0][1]: " << scan[0][1] << "\n";

    return 0;
}
```

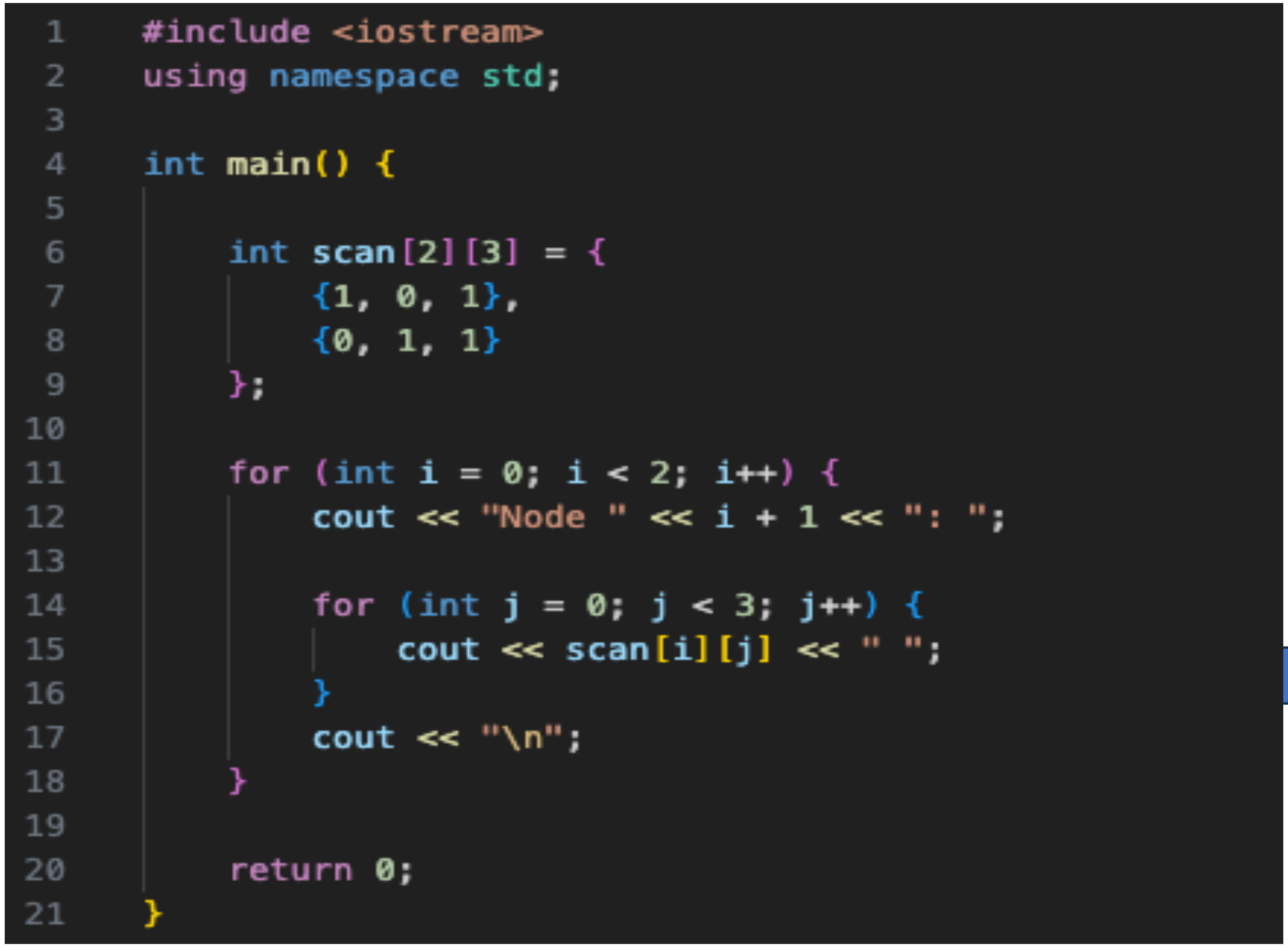
This is how systems map networks.

Output

?

Part 6: Looping Through 2D Arrays

❖ Example: Display Scan Results



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

int main() {

    int scan[2][3] = {
        {1, 0, 1},
        {0, 1, 1}
    };

    for (int i = 0; i < 2; i++) {
        cout << "Node " << i + 1 << ": ";

        for (int j = 0; j < 3; j++) {
            cout << scan[i][j] << " ";
        }
        cout << "\n";
    }

    return 0;
}
```

Nested loops scan networks level by level.

Output

?

Part 7: Challenge Exercise 1

Scenario: A security system records the **number of suspicious events** detected each hour.

- Write a C++ program that:
  - Uses a **one-dimensional array** of size **10**
  - Reads the number of suspicious events for each hour
- Calculates:
  - Total number of events
  - Average number of events
- Displays a **security status**:
  - ✓ If average > 5 → High Threat Level
  - ✓ Else → Normal Activity

Part 7: Challenge Exercise 2

Scenario: A cybersecurity system scans **multiple servers** and checks **multiple ports**.

1 → Port is open (risk)

0 → Port is closed (safe)

Write a C++ program that:

1. Uses a **2D array** of size **3 × 4**
   - 3 servers
   - 4 ports per server
2. Reads scan results from the user
3. For each server:
   - Counts how many ports are open
   - Displays the count
4. If any server has **more than 2 open ports**, display:

Alert: High Risk Server Detected

Summary

- Arrays store **multiple values**
- Index starts from **0**
- 1D arrays → lists
- 2D arrays → tables
- Loops process arrays efficiently
- Cybersecurity systems depend on arrays

Thank You