

Introduction to Arrays in C++

From Variables to Collections

Mr. Gullo

July 20, 2025

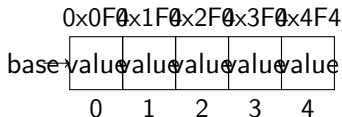
Learning Objectives

After this lesson, you will be able to:

- Understand arrays as contiguous memory structures
- Declare and initialize arrays effectively
- Perform common array operations
- Implement array-based solutions
- Avoid common array pitfalls

Array Memory Layout

- Contiguous memory allocation
- Each element occupies fixed space
- Direct access via index



Array Creation

Declaration Methods

```
1 // Method 1: Declaration with size
2 int temperatures[7]; // Uninitialized array
3
4 // Method 2: Declaration with initialization
5 int scores[5] = {95, 88, 76, 90, 85};
6
7 // Method 3: Size inference
8 int fibonacci[] = {1, 1, 2, 3, 5, 8, 13};
9
10 // Method 4: Partial initialization
11 int values[5] = {0}; // Sets all to 0
```

Working with Arrays

```
1 int numbers[] = {10, 20, 30, 40, 50};
2
3 // Reading elements
4 int first = numbers[0];      // Get first (10)
5 int last = numbers[4];      // Get last (50)
6
7 // Modifying elements
8 numbers[2] = 35;            // Change middle element
9 numbers[4] += 5;            // Increment last element
10
11 // Common mistake: bounds
12 // numbers[5] = 60;         // ERROR: Out of bounds!
```

Array Traversal Patterns

```
1 int data[] = {1, 2, 3, 4, 5};
2 int size = 5;
3
4 // Forward traversal
5 for(int i = 0; i < size; i++) {
6     cout << data[i] << " ";
7 }
8
9 // Reverse traversal
10 for(int i = size - 1; i >= 0; i--) {
11     cout << data[i] << " ";
12 }
13
14 // Skip pattern (every second element)
15 for(int i = 0; i < size; i += 2) {
16     cout << data[i] << " ";
17 }
```

Example: Temperature Analysis

```
1 // Find average daily temperature
2 double getAverageTemp(int temps[], int days) {
3     double sum = 0;
4     for(int i = 0; i < days; i++) {
5         sum += temps[i];
6     }
7     return sum / days;
8 }
9
10 // Find temperature range
11 void getTempRange(int temps[], int days,
12                  int& min, int& max) {
13     min = max = temps[0];
14     for(int i = 1; i < days; i++) {
15         if(temps[i] < min) min = temps[i];
16         if(temps[i] > max) max = temps[i];
17     }
18 }
```

Example: Data Processing

```
1 // Count occurrences in an array
2 int countValue(int arr[], int size, int target) {
3     int count = 0;
4     for(int i = 0; i < size; i++) {
5         if(arr[i] == target) count++;
6     }
7     return count;
8 }
9
10 // Check if array is sorted
11 bool isSorted(int arr[], int size) {
12     for(int i = 1; i < size; i++) {
13         if(arr[i] < arr[i-1]) return false;
14     }
15     return true;
16 }
```


Array Best Practices

Common Pitfalls

- Array bounds violations
- Off-by-one errors in loops
- Uninitialized array access
- Forgetting array size limits

Best Practices

- Always track array size
- Initialize arrays when declared
- Use bounds checking
- Consider using `std::array` when possible

Key Takeaways

Core Concepts

- Contiguous memory
- Zero-based indexing
- Fixed size
- Type consistency

Operations

- Element access
- Array traversal
- Data processing
- Bounds checking