OPENTECHZ

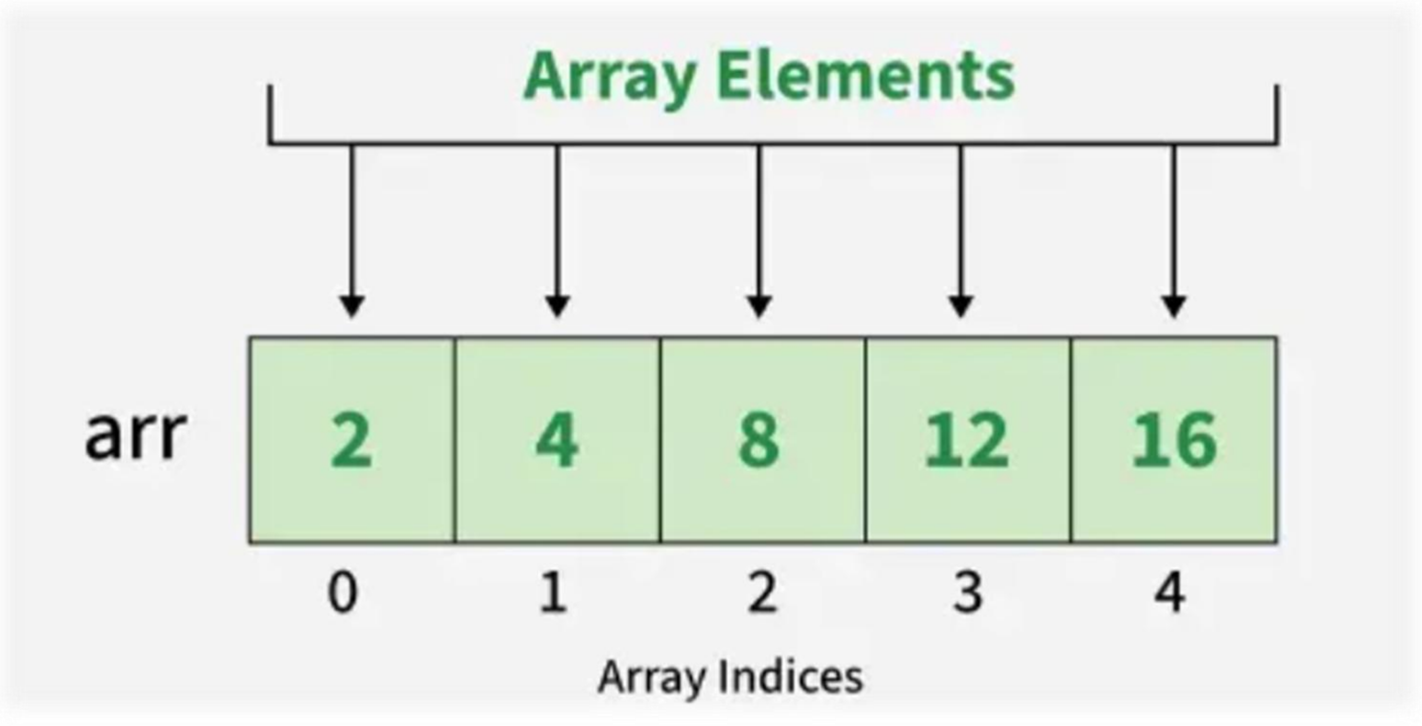
A dream to realize your dream

Array

Opentechz Pvt Ltd .
By Parthasarathi Swain

What is an Array?

- An array is a **fixed-size sequential collection** of elements of the **same data type**.
- Each element is stored in **continuous memory**.
- Accessed using **index numbers (0 to size-1)**.
- Supports **random access** using index.
- Example: int marks[5]; stores 5 integers.



Why Use Arrays?

- ✓ Avoids declaring multiple variables.
- ✓ Easy to **manage large amounts of data**.
- ✓ Useful in **loops, searching, sorting, matrices**, etc.
- ✓ Arrays reduce **code length and complexity**.
- ✓ Data stored in a structured manner.

Without Array:

```
int m1=10;
int m2=20;
int m3=30;
int m4=40;
int m5=50;
```

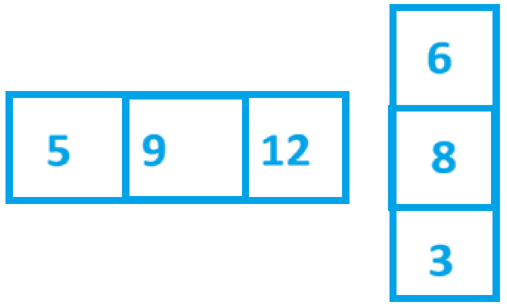
With Array:

```
int  arr[5] =  {10, 20, 30, 40, 50};
```

Types of Arrays in C++

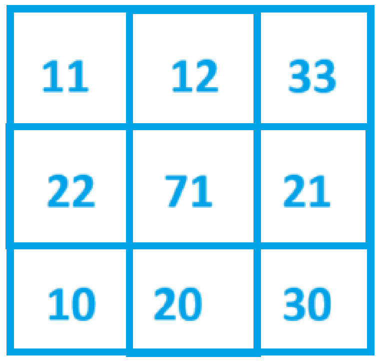
One-Dimensional Array (1D)

- Stores data in a **single row**.
- Syntax: int arr[5];
- Example: int marks[5] = {90, 85, 75, 88, 92};



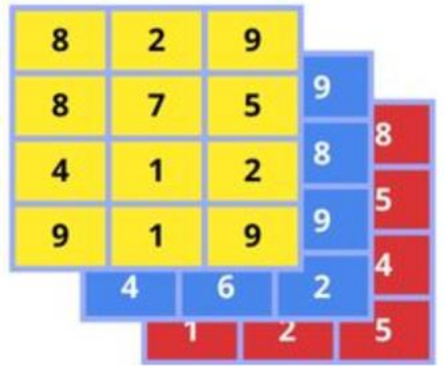
Two-Dimensional Array (2D)

- Stores data in **rows and columns** (like a table).
- Syntax: int arr[3][3];
- Example: int matrix[2][2] = {{1, 2}, {3, 4}};



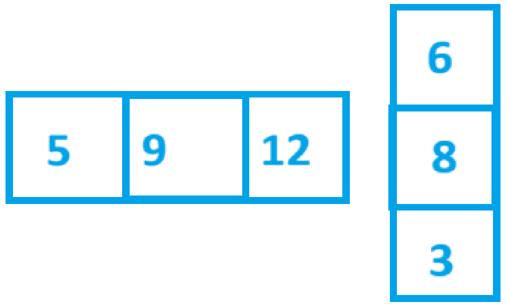
Multidimensional Array

- Arrays with **3 or more dimensions**.
- Syntax: int arr[2][3][4];
- Example: Often used in **scientific or 3D data**.



One-Dimensional Array (1D)

A **1D (One-Dimensional) Array** in C++ is a collection of elements of the **same data type**, stored in **contiguous memory locations**, and accessed using an **index**.



Declaration Only:

```
int arr[5]; // Uninitialized array of 5 integers
```

Declaration with Initialization:

```
int arr[5] = {10, 20, 30, 40, 50}; // Initialize with values
```

Let Compiler Count Size

```
int arr[] = {5, 10, 15, 20}; // Compiler sets size = 4
```

Partial Initialization:

```
int arr[5] = {1, 2}; // Remaining elements = 0
```

Important Points

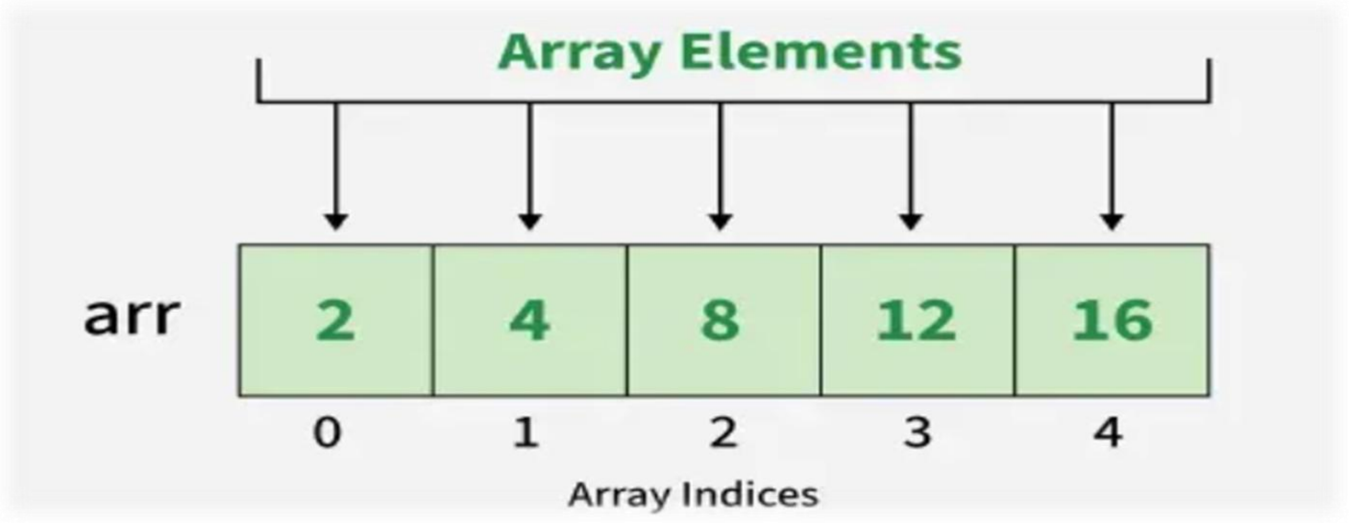
Array index starts from **0 to size-1**.

All elements must be of the **same data type**.

Stored in **contiguous memory**.

Array name is a **pointer to the first element**.

Access using: arr[index].



Size of Array

Method 1: Using sizeof() (Works in all C++ versions)

```
Size = sizeof(arr) / sizeof(arr[0])
```

```
int arr[] = {10, 20, 30, 40, 50};
```

- sizeof(arr) → 5 elements × 4 bytes each = 20 bytes
- sizeof(arr[0]) → size of first element (int) = 4 bytes
- So, sizeof(arr) / sizeof(arr[0]) = 20 / 4 = 5

Method 2: Using std::size() (C++17 and above)

```
Size = std::size(arr);
```

```
int arr[] = {10, 20, 30, 40, 50};

int size = std::size(arr);   // size=5
```

1. Input & Print Elements

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

int main() {
    int arr[5];
    cout << "Enter 5 elements: ";
    for(int i = 0; i < 5; i++) {
        cin >> arr[i];
    }

    cout << "Array elements are: “<<endl;
    for(int i = 0; i < 5; i++) {
        cout << arr[i] << " ";
    }

    return 0;
}
```

```
Array elements are:
10 20 30 40 50
```

2. Sum of Elements

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

int main() {
    int arr[5] = {10, 20, 30, 40, 50}, sum = 0;

    for(int i = 0; i < 5; i++) {
        sum += arr[i];
    }

    cout << "Sum = " << sum;
    return 0;
}
```

```
Sum = 150
```