Arrays

Module 3.1

Array definition

- An **array** is a collection of elements of the same type placed in contiguous memory locations.
- It allows you to store multiple values under a single name and access them using an index.

Declaration:

data_type array_name[size];

int numbers[5]; // declares an array of 5 integers

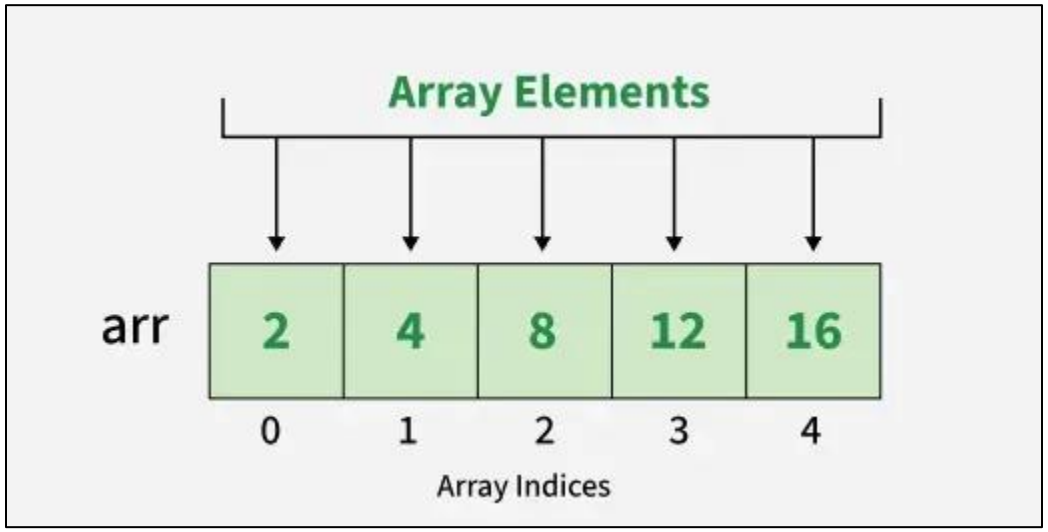
Initialization:

```
int numbers[5] = {2, 4, 8, 12, 16};
```

or

```
numbers[0] = 2;
numbers[1] = 4;
```

Explanation



- **int arr[5]** declares an array of 5 integers.
- The elements are initialized with {2, 4, 8, 12, 16}.
- The for loop is used to iterate over the array and print each element.
- Array indices in C++ start from 0, so arr[0] refers to the first element, and arr[4] refers to the last one in this case.

Different ways of initialization of array

🔹 Different Ways to Initialize Arrays

- **Without Initialization**

```
int arr[5];   // contains garbage values
```

- **Full Initialization**

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

- **Partial Initialization**

```
int arr[5] = {10, 20};
// arr = {10, 20, 0, 0, 0}
```

Different ways of initialization of array...

- **Compiler-Sized Initialization**

```
int arr[] = {1, 2, 3, 4};  // size = 4
```

- **All Zeros Initialization**

```
int arr[5] = {0};
// arr = {0, 0, 0, 0, 0}
```

Accessing Elements:

- cout << numbers[2]; // prints 8
- numbers[4] = 50; // updates 5th element

Features

- Fixed size (size must be known at compile time).
- Efficient for sequential storage and iteration using loops.
- All elements must be of the **same data type**

Print elements in an array

```
#include <iostream>
using namespace std;
int main() {
    // Declare and initialize an array of 5 integers
    int numbers[5] = {10, 20, 30, 40, 50};

    // Print all elements of the array
    cout << "Array elements are: ";
    for(int i = 0; i < 5; i++) {
        cout << numbers[i] << " ";
    }
    cout << endl;

    return 0;
}
```

program to print alternate numbers from a one-dimensional array:

```
#include <iostream>
using namespace std;
int main() {
    int numbers[6] = {10, 20, 30, 40, 50, 60};
    cout << "Alternate numbers in the array are: ";
    for(int i = 0; i < 6; i += 2) {
        cout << numbers[i] << " ";
    }
    cout << endl;
    return 0;
}
```

Find the Largest Element in an Array

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

int main() {
    int n;
    cout << "Enter number of elements: ";
    cin >> n;

    int arr[100];
    cout << "Enter " << n << " elements: ";
    for (int i = 0; i < n; i++)
        cin >> arr[i];

    int largest = arr[0];
    for (int i = 1; i < n; i++) {
        if (arr[i] > largest)
            largest = arr[i];
    }

    cout << "Largest element: " << largest << endl;
    return 0;
}
```

```
Enter number of elements: 5
Enter 5 elements: 21 56 78 34 66
Largest element: 78
```

Find the Sum and Average of element in an array

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

int main() {
    int n;
    cout << "Enter number of elements: ";
    cin >> n;

    int arr[100];
    cout << "Enter elements: ";
    for (int i = 0; i < n; i++)
        cin >> arr[i];

    int sum = 0;
    for (int i = 0; i < n; i++)
        sum += arr[i];

    cout << "Sum = " << sum << endl;
    cout << "Average = " << (float)sum / n << endl;
    return 0;
}
```

```
Enter number of elements: 5
Enter elements: 1 2 3 4 5
Sum = 15
Average = 3
```

Find the count of odd and even elements in an array

Search an element in an array

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

int main() {
    int n, key, found = 0;
    cout << "Enter number of elements: ";
    cin >> n;

    int arr[100];
    cout << "Enter elements: ";
    for (int i = 0; i < n; i++)
        cin >> arr[i];

    cout << "Enter element to search: ";
    cin >> key;

    for (int i = 0; i < n; i++) {
        if (arr[i] == key) {
            cout << "Element found at position " << i + 1 << endl;
            found = 1;
            break;
        }
    }

    if (!found)
        cout << "Element not found." << endl;
    return 0;
}
```

```
Enter number of elements: 5
Enter elements: 2 7 9 4 6
Enter element to search: 9
Element found at position 3
```

Multidimensional array

A **multidimensional array** is an array of arrays.

The most common is the **two-dimensional (2D) array**, which can be visualized like a table (rows × columns).

Declaration

- data_type array_name[rows][columns];
- int matrix[3][3]; // 3 rows and 3 columns

Initialization

```
int matrix[2][3] = {
    {1, 2, 3},
    {4, 5, 6}
};
```

Accessing Elements

- cout << matrix[0][1]; // prints 2

Demonstrate a Two-Dimensional (2D) Array

```
#include <iostream>
using namespace std;
int main() {
    // Declare and initialize a 2D array
    int matrix[2][3] = {
        {1, 2, 3},
        {4, 5, 6}
    };
    // Print the 2D array
    cout << "Matrix elements are:" << endl;
    for(int i = 0; i < 2; i++) {
        for(int j = 0; j < 3; j++) {
            cout << matrix[i][j] << " ";
        }
        cout << endl; // new line after each row
    }
    return 0;
}
```

Matrix elements are:

1 2 3

4 5 6

Find the sum of all elements in a matrix

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

int main() {
    int a[10][10], rows, cols, sum = 0;
    cout << "Enter rows and columns: ";
    cin >> rows >> cols;

    cout << "Enter matrix elements:\n";
    for (int i = 0; i < rows; i++)
        for (int j = 0; j < cols; j++) {
            cin >> a[i][j];
            sum += a[i][j];
        }

    cout << "Sum of all elements: " << sum;
    return 0;
}
```

```
Enter rows and columns: 3 3
Enter matrix elements:
21 56 45
45 77 8
23 56 77
Sum of all elements: 408
```

Smallest element in the matrix

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

int main() {
    int a[10][10], rows, cols, smallest;
    cout << "Enter rows and columns: ";
    cin >> rows >> cols;

    cout << "Enter matrix elements:\n";
    for (int i = 0; i < rows; i++)
        for (int j = 0; j < cols; j++)
            cin >> a[i][j];

    smallest = a[0][0];
    for (int i = 0; i < rows; i++)
        for (int j = 0; j < cols; j++)
            if (a[i][j] < smallest)
                smallest = a[i][j];

    cout << "Smallest element in matrix: " << smallest;
    return 0;
}
```

```
Enter rows and columns: 2 3
Enter matrix elements:
23 45 56
1 9 42
Smallest element in matrix: 1
```

Find the row and column sum of the matrix

1. Find the Error

```
int arr[5];
for (int i = 1; i <= 5; i++)
    cin >> arr[i];
```

1. Find the Error

```
int arr[5];
for (int i = 1; i <= 5; i++)
    cin >> arr[i];
```

- ❌ Error: Array index starts at 0. arr[5] is out of bounds.
- ✅ Fix: for (int i = 0; i < 5; i++) cin >> arr[i];

2. Find the error

```
int arr[5] = {10, 20, 30, 40, 50};
for (int i = 0; i <= 5; i++)
    cout << arr[i];
```

2. Find the error

```
int arr[5] = {10, 20, 30, 40, 50};
for (int i = 0; i <= 5; i++)
    cout << arr[i];
```

- ❌ Error: Accessing arr[5] (invalid index).
- ✅ Fix: Use i < 5 instead of i <= 5.

3. Fix the solution for wrong output

```
int arr[5] = {5, 10, 15, 20, 25};
int sum;
for (int i = 0; i < 5; i++)
    sum += arr[i];
```

3. Fix the solution for wrong output

```
int arr[5] = {5, 10, 15, 20, 25};
int sum;
for (int i = 0; i < 5; i++)
    sum += arr[i];
```

- ❌ wrong output: sum is not initialized to zero
- ✅ Fix: sum=0

4. Partial Initialization

- int arr[5] = {1, 2, 3};
- for (int i = 0; i < 5; i++)
- cout << arr[i] << " ";

- ✅ Output: 1 2 3 0 0
- 💡 Uninitialized elements default to 0.

5. Missing Dimension Size

- int arr[][] = {{1,2,3},{4,5,6}};

- ❌ Error: Missing column size.
- ✅ Fix: int arr[2][3] = {{1,2,3},{4,5,6}};

6. Logical Error in Row Sum

```
int a[3][3] = {{1,2,3},{4,5,6},{7,8,9}};
for (int i = 0; i < 3; i++) {
    int sum = 0;
    for (int j = 0; j < 3; j++)
        sum = a[i][j];
    cout << sum;
}
```

6. Logical Error in Row Sum

```
int a[3][3] = {{1,2,3},{4,5,6},{7,8,9}};
for (int i = 0; i < 3; i++) {
    int sum = 0;
    for (int j = 0; j < 3; j++)
        sum = a[i][j];
    cout << sum;
}
```

- ❌ Error: Overwrites sum instead of adding.
- ✅ Fix: sum += a[i][j];