Array in C++

---

1. Introduction to Arrays in C++

- Arrays in C++ are a collection of elements of the same type stored in contiguous memory locations.
- They are used to store multiple values in a single variable, which can be accessed using indices.

---

2. Features of Arrays in C++

1. **Fixed Size**: Arrays have a predefined size that cannot be altered at runtime.
2. **Homogeneous Data**: All elements in an array are of the same data type.
3. **Zero-based Indexing**: The first element of an array has an index of 0.

---

3. Syntax for Arrays

```
// Declaration
int arr[5]; // An array of 5 integers

// Initialization
int arr[5] = {1, 2, 3, 4, 5};

// Accessing Elements
cout << arr[0]; // Access the first element
```

---

4. Introduction to `<array>` in C++

- `<array>` is a part of the Standard Template Library (STL) introduced in C++11.
- Provides a safer and more feature-rich alternative to C-style arrays.

---

5. Key Differences Between `<array>` and C-style Arrays

Feature	C-style Array	`<array>` in C++
Bounds Checking	Not Available	Available through `.at()`
Size Access	Not Available	Available through `.size()`
Functionality	Limited	Richer, includes methods like `.fill()` and `.swap()`
Safety	Less Safe	Safer with standard library support

---

6. Basic Functions of `<array>` in C++

6.1 `.at()`

- Accesses an element with bounds checking.

```
arr.at(index);
```

6.2 `.size()`

- Returns the total number of elements in the array.

```
arr.size();
```

6.3 `.front()` and `.back()`

- Accesses the first and last elements of the array.

```
arr.front(); // First element
arr.back();  // Last element
```

6.4 `.fill(value)`

- Fills all elements of the array with the specified value.

```
arr.fill(value);
```

6.5 `.swap(other_array)`

- Swaps the contents of the array with another array of the same type and size.

```
arr1.swap(arr2);
```

---

7. Benefits of Using `<array>`

1. **Safety**: Bounds checking helps prevent runtime errors.
2. **Readability**: Easier to understand and manage.
3. **Rich API**: Provides a range of useful functions.
4. **Standardization**: Part of STL, ensuring consistency and reliability.

---

8. Summary

- `<array>` in C++ offers a modern, feature-rich alternative to traditional arrays.
- It is part of the STL, providing enhanced safety, functionality, and ease of use.
