Programming Fundamentals with C++

Lecture 12 – Arrays

Dr. Muhammad Sajjad

RA: Asad Ullah

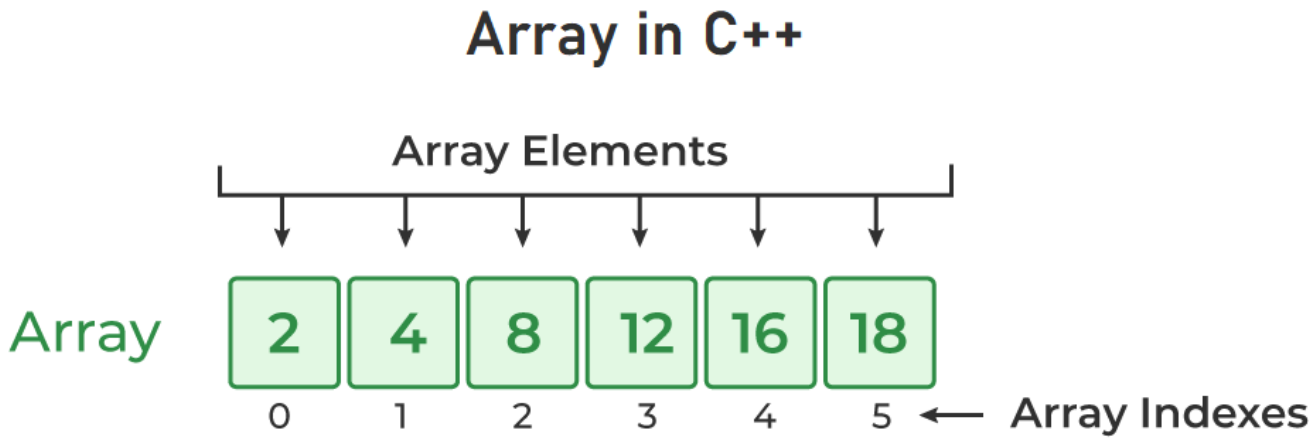
Overview

- **Arrays in C++**
  - What is an Array?
  - Why Do We Need Arrays?
  - Properties of Arrays
  - Array Declaration
  - Array Initialization
  - Accessing Elements
  - Updating Elements
  - How Arrays Work in Memory
  - Different Datatypes Arrays

Arrays in C++

What is an Array?

- In C++, an array is a data structure that is used to store multiple values of similar data types in a contiguous memory location.
- **For example**, if we have to store the marks of 4 or 5 students then we can easily store them by creating 5 different variables but what if we want to store marks of 100 students or say 500 students then it becomes very challenging to create that numbers of variable and manage them. Now, arrays come into the picture that can do it easily by just creating an array of the required size.



Arrays in C++

Why Do We Need Arrays?

Consider a scenario where you need to store the marks of 5 students:

```
int mark1 = 90;
int mark2 = 85;
int mark3 = 78;
int mark4 = 92;
int mark5 = 88;
```

This approach becomes cumbersome as the number of students increases. Instead, we can use an **array**:

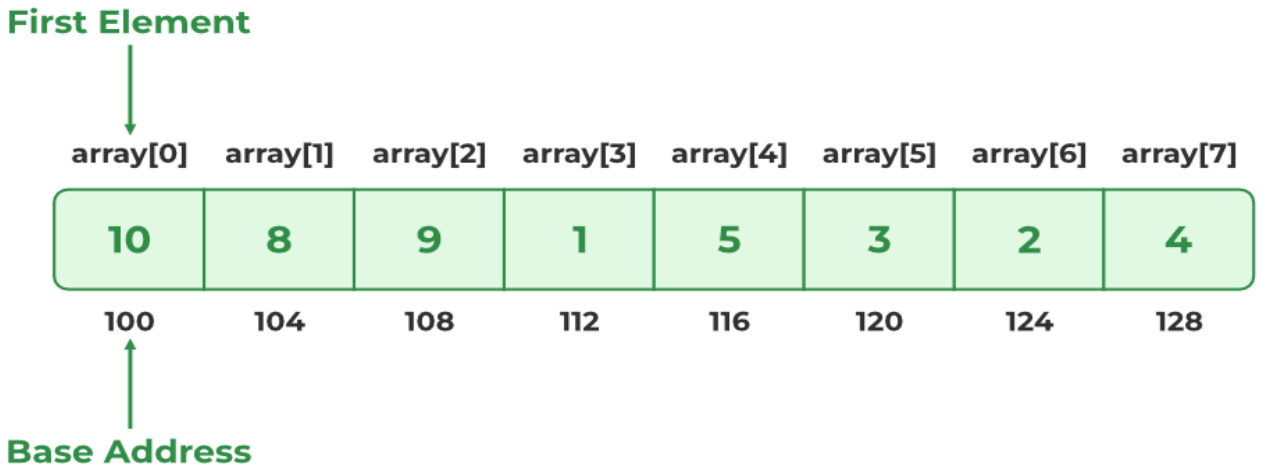
```
int marks[5] = {90, 85, 78, 92, 88};
```

With an array, we can manage a collection of data efficiently.

Arrays in C++

Properties of Arrays

- An Array is a collection of data of the same data type, stored at a contiguous memory location.
- Indexing of an array starts from **0.** It means the first element is stored at the 0th index, the second at 1st, and so on.
- Elements of an array can be accessed using their indices.
- Once an array is declared its size remains constant throughout the program.
- An array can have multiple dimensions.



Arrays in C++

Array Declaration

In C++, we can declare an array by simply specifying the data type first and then the name of an array with its size.

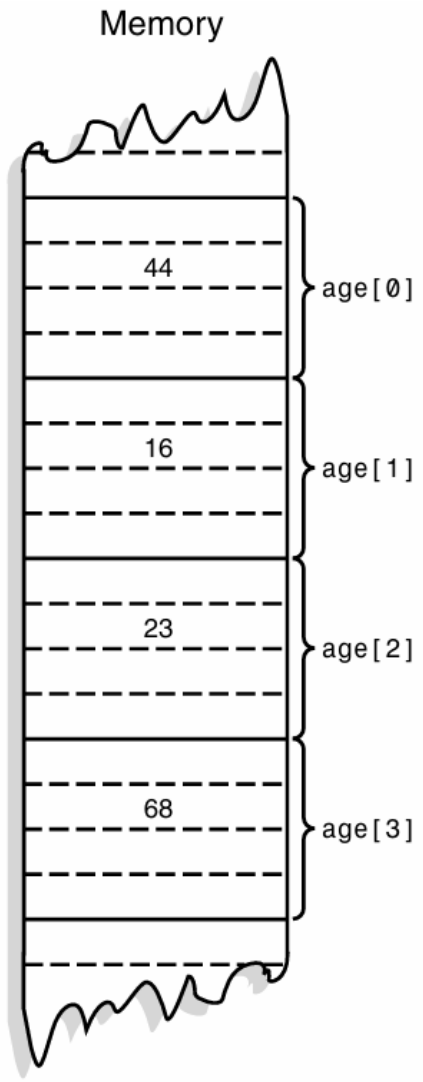
```
data_type array_name[Size_of_array];
```

Example

```
int age[4];
```

Here,

- **int:** It is the type of data to be stored in the array. We can also use other data types such as char, float, and double.
- **age:** It is the name of the array.
- **4:** It is the size of the array which means only 5 elements can be stored in the array.



Arrays in C++

Initialization of Array

In C++, we can initialize an array in many ways but we will discuss some most common ways to initialize an array.

We can initialize an array at the time of declaration or after declaration.

1. Initialize Array with Values
2. Initialize Array with Values and without Size
3. Initialize Array after Declaration (Using Loops)
4. Initialize an array partially

Arrays in C++

Initialization of Array

1. Initialize Array with Values

We have initialized the array with values. The values enclosed in curly braces '{}' are assigned to the array. Here, 55 is stored in marks[0], 75 in marks[1], and so on. Here the size of the array is 5.

```
int marks[5] = {55,75,61,80,78};
```

2. Initialize Array with Values and without Size

We have initialized the array with values but we have not declared the length of the array, therefore, the length of an array is equal to the number of elements inside curly braces.

```
int marks[ ] = {66,75,88,60};
```

Arrays in C++

Initialization of Array

3. Initialize Array after Declaration (Using Loops)

We have initialized the array using a loop after declaring the array. This method is generally used when we want to take input from the user or we cant to assign elements one by one to each index of the array. We can modify the loop conditions or change the initialization values according to requirements.

```
int marks[5];
for (int i = 0; i < 5: i++) {
        cin>>marks[i];
}
```

4. Initialize an array partially

Here, we have declared an array 'partialArray' with size '5' and with values '1' and '2' only. So, these values are stored at the first two indices, and at the rest of the indices '0' is stored.

```
int partialArray [ 5 ] = {1,2};
```

Arrays in C++

Accessing an Element of an Array

Elements of an array can be accessed by specifying the name of the array, then the index of the element enclosed in the array subscript operator []. For example, arr[i].

```
#include <iostream>
using namespace std;
int main() {
    int arr[3];
    // Inserting elements in an array
    arr[0] = 10;
    arr[1] = 20;
    arr[2] = 30;
    // Accessing and printing elements of the array
    cout << "arr[0]: " << arr[0] << endl;
    cout << "arr[1]: " << arr[1] << endl;
    cout << "arr[2]: " << arr[2] << endl;
    return 0;
}
```

- Indices start from 0 (i.e., the first element is array_name[0]).
- The last element has an index of size - 1.

Arrays in C++

Update Array Element

To update an element in an array, we can use the index which we want to update enclosed within the array subscript operator and assign the new value.

```
arr[index] = new_value;
```

How Arrays Work in Memory

Arrays store elements in contiguous memory locations. For example:

```
int numbers[3] = {10, 20, 30};
```

Index	Address	Value
0	0x100	10
1	0x104	20
2	0x108	30

Here, the size of an int is 4 bytes, so each element is stored 4 bytes apart.

Arrays in C++

Different Datatypes Arrays

In C++, arrays can store elements of a specific data type. The type of array depends on the type of elements it holds. Below is an explanation of different array data types in C++:

1. Integer Array (int[ ]): Stores whole numbers.

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

2. Floating-Point Array (float[ ] or double[ ]): Stores numbers with fractional parts.

```
float heights[ ] = {5.5f, 6.2f, 4.9f};
```

```
double distances[ ] = {10.5, 20.75, 30.1};
```

3. Boolean Array (bool[ ]): Stores true or false values.

```
bool isPassed[ ] = {true, false, true, false};
```

4. Character Array (char[ ]): Stores characters. It can also store strings when null-terminated.

```
char vowels[ ] = {'a', 'e', 'i', 'o', 'u'};
char name[ ] = "John";      // same as : char name[ ] = {‘J’, ‘o’, ‘h’, ‘n’, ‘\0’};
```

Arrays in C++

Arrays Programs

Programs without Loops

Programs with Loops

Real-Life Scenarios

Thank You