CSE 06131223 ♦ CSE 06131224

Structured Programming

Lecture 11

Array in C (1)

Prepared by

Md. Mijanur Rahman, Prof. Dr.

Dept. of Computer Science and Engineering

Jatiya Kabi Kazi Nazrul Islam University, Bangladesh

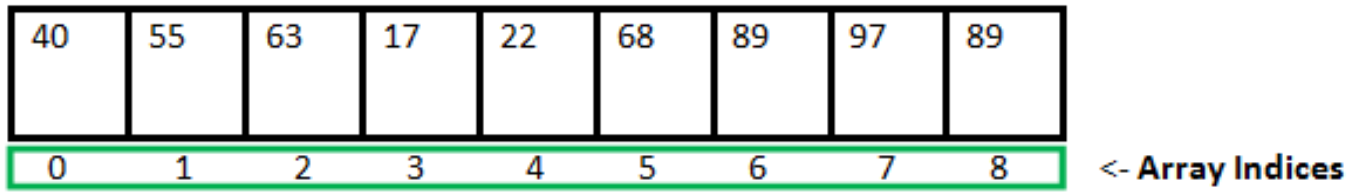
www.mijanrahman.com

Contents

ARRAY IN C

C Array

- An array is defined as the collection of similar type of data items stored at contiguous memory locations. Arrays are the derived data type in C programming language which can store the primitive type of data such as int, char, double, float, etc.
- **Array is a collection of elements of same type referred using a unique name. Each element of an array is stored in memory locations indexed from 0 through n number of its elements.**
- The array is the simplest data structure where each data element can be randomly accessed by using its index number.



Array Length = 9
First Index = 0
Last Index = 8

Properties of Array

- **The array contains the following properties:**
  - **Each element of an array is of same data type and carries the same size, i.e., int = 4 bytes.**
  - **Elements of the array are stored at contiguous memory locations,** where the first element is stored at the smallest memory location.
  - **Elements of the array can be randomly accessed** since we can calculate the address of each element of the array with the given base address and the size of the data element.

Advantage of C Array

1. **Code Optimization**: Less code to the access the data.
2. **Ease of traversing**: By using the for loop, we can retrieve the elements of an array easily.
3. **Ease of sorting**: To sort the elements of the array, we need a few lines of code only.
4. **Random Access**: We can access any element randomly using the array.

Disadvantage of C Array

1. **Fixed Size:** Whatever size, we define at the time of declaration of the array, we can't exceed the limit. Unlike a linked list, an array in C is not dynamic.
2. Insertion and deletion of elements can be costly since the elements are needed to be managed in accordance with the new memory allocation.

Why do we need arrays?

- We can use normal variables (v1, v2, v3, ..) when we have a small number of objects, but if we want to store a large number of instances, it becomes difficult to manage them with normal variables.
- The idea of an array is to represent many instances in one variable.

Declaration of C Array

- We can declare an array in the c language in the following way.

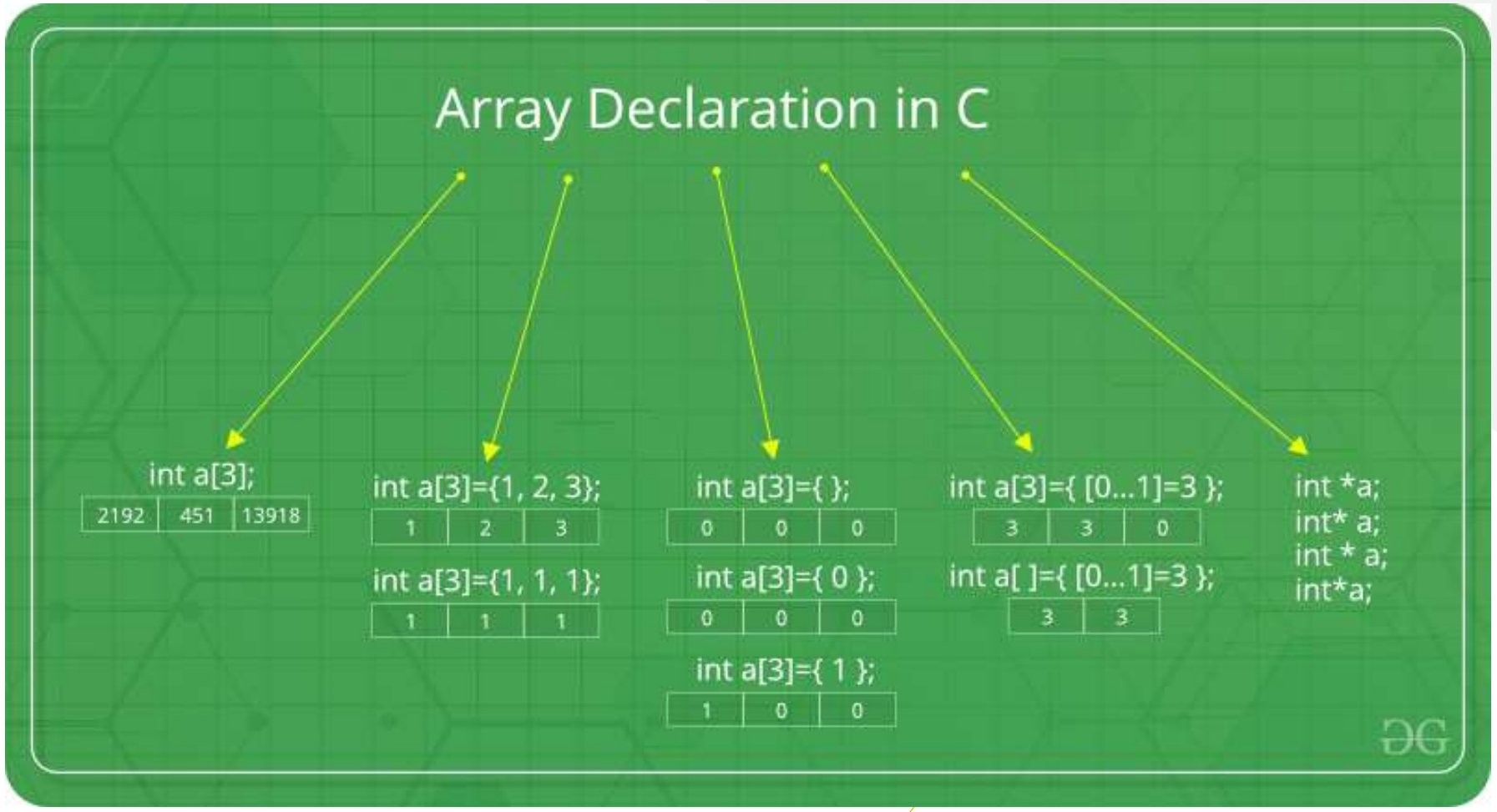
  data-type arrayName[array-size];

- Now, let us see the example to declare the array.

  int marks[5];

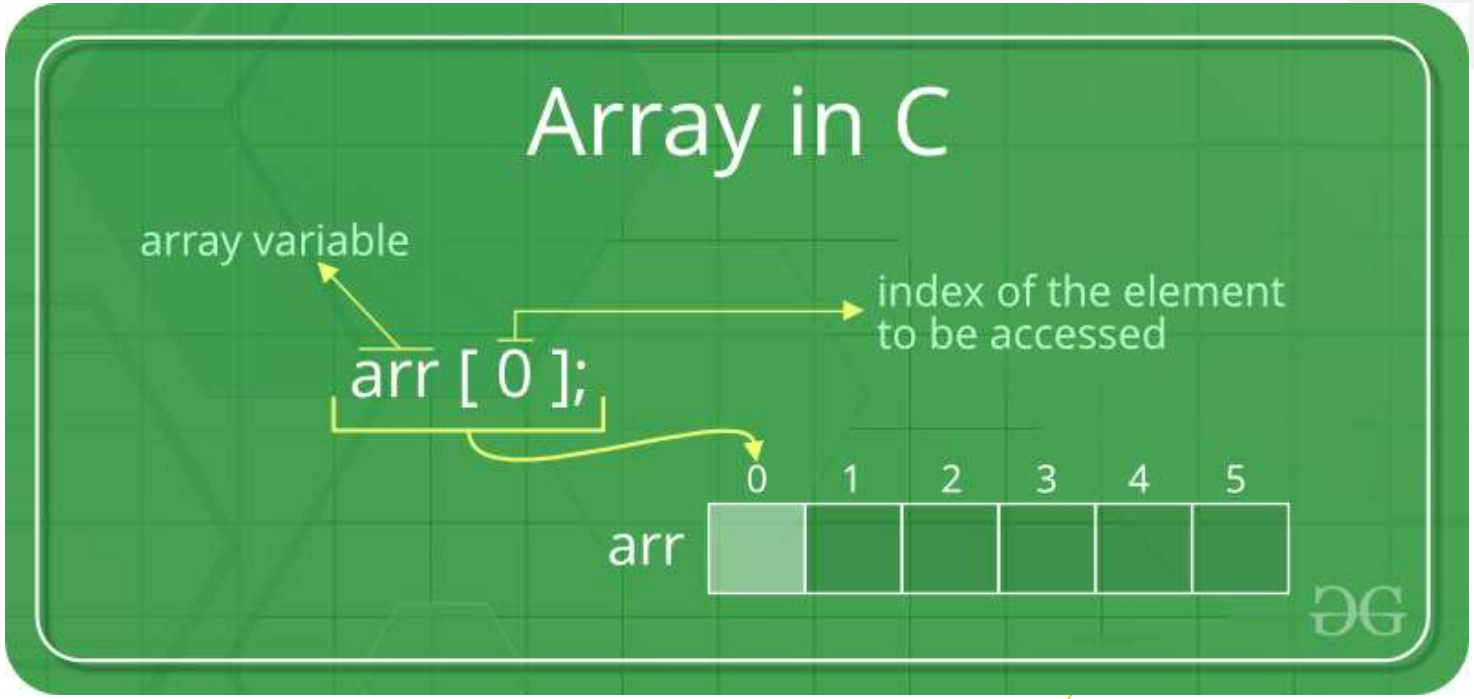
- Here, int is the *data-type*, marks are the *arrayName*, and 5 is the *array-size*.

Declaration of C Array



Accessing Array Elements

- Array elements are accessed by using an integer index. Array index starts with 0 and goes till size of array minus 1.
- Name of the array is also a pointer to the first element of array.



Accessing Array Elements

```c
int main(){
    int arr[5];
    arr[0] = 5;
    arr[3 / 2] = 2; // this is same as arr[1] = 2
    arr[2] = -10;
    arr[3] = arr[0];
    arr[4] = arr[0] + arr[1];

printf("The given array elements:\n");
printf("%d %d %d %d %d", arr[0], arr[1], arr[2], arr[3], arr[4]);

return 0;
}
```

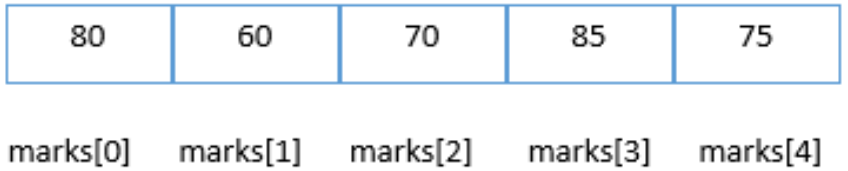
Terminal

```
The given array elements:
5 2 -10 5 7
```

Initialization of C Array

- **The simplest way to initialize an array is by using the index of each element.** We can initialize each element of the array by using the index.
- Consider the following example:

```
marks[0]=80;//initialization of array
marks[1]=60;
marks[2]=70;
marks[3]=85;
marks[4]=75;
```



Initialization of Array

Initialization of C Array

- Initialization of arrays in C suffers two drawbacks:
  1. There is no convenient way to initialize only selected elements.
  2. There is no shortcut method for initializing a large number of array elements.

Initialization of C Array

- There are two main ways to initialize arrays in C:

  1. Initialization during Declaration 2. Initialization after Declaration

1. Initialization during Declaration:

- This is the most common way to set initial values for array elements. It provides a comma-separated list of values enclosed in curly braces { } along with the array declaration. The number of elements in the list determines the size of the array.
- For example:

```
int numbers[5] = {10, 20, 30, 40, 50}; // Array with size 5 and initial values
char greetings[] = {'H', 'e', 'l', 'l', 'o', '!'}; // Character array initialized
```

Initialization of C Array

2. Initialization after Declaration:

- Here, we can declare the array with its size and then assign values to individual elements later in the code. This is useful when we don't have the initial values readily available or need to calculate them dynamically.

- For example:

```
int scores[3]; // Array of integers with size 3 (not initialized)
:
scores[0] = 85;
scores[1] = 92;
scores[2] = 78; // Assigning values to each element
```

Initialization of C Array

Array Initialization after Declaration:

```c
#include<stdio.h>
int main(){
  int i;
  int marks[5]; //declaration of array
  marks[0]=80; //initialization of array
  marks[1]=60;
  marks[2]=70;
  marks[3]=85;
  marks[4]=75;

  //Traversal of array
  printf("Elements of Array:\n");
  for(i=0;i<5;i++){
    printf("%d \n", marks[i]);
  }
  return 0;
}
```

Terminal

```
Elements of Array:
80
60
70
85
75
```

Initialization of C Array

Array Declaration and Initialization :

```
Terminal

Elements of Array:
20
30
40
50
60
```

```c
#include<stdio.h>
int main(){
   //declaration and initialization
   int marks[5] = {20,30,40,50,60};

   //Traversal of array
   printf("Elements of Array:\n");
   for(int i=0; i<5; i++){
      printf("%d \n", marks[i]);
    }
 return 0;
}
```

Initialization of C Array

Initializing all elements with the same value:

```c
#include <stdio.h>
int main() {
  int numbers[10] = {0}; // All elements initialized to 0

  printf("Initial Values:\n");
  for (int i = 0; i < 10; i++) {
    printf("%d ", numbers[i]);
  }

  printf("\n");
  return 0;
}
```

Terminal

```
Initial Values:
0 0 0 0 0 0 0 0 0 0
```

Initialization of C Array

Initializing specific elements:

```c
#include <stdio.h>

int main() {
  // Initialize first 3 elements
  char word[8] = {'C', 'O', 'M', 'P', 'U', 'T', 'E', 'R'};

  word[8] = '\0'; // Add null terminator manually

  printf("Given Word: %s\n", word);
  return 0;
}
```

Terminal

```
Given Word: COMPUTER
```

?

THE END