SESSION-6

Array:

An **array** is a linear data structure that stores a fixed-size sequence of elements of the same data type in contiguous memory locations. Each element can be accessed directly using its index, which allows for efficient retrieval and modification.

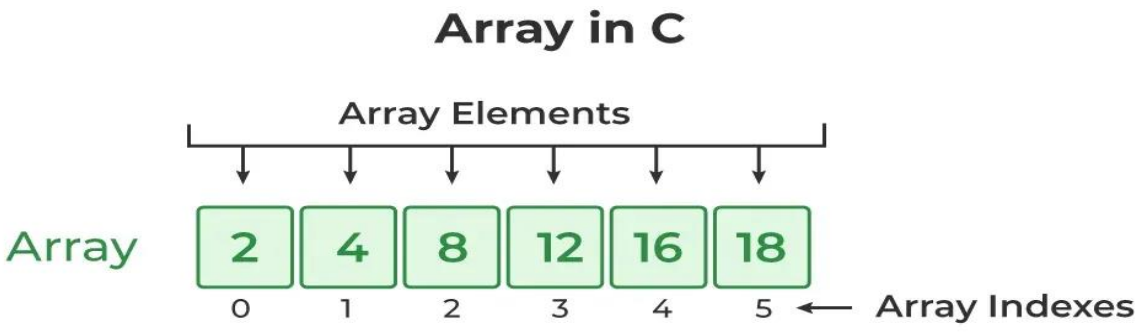
Example:

```
#include <stdio.h>
int main() {
int arr[] = {2, 4, 8, 12, 16, 18};
int n = sizeof(arr)/sizeof(arr[0]);
// Printing array elements
for (int i = 0; i < n; i++)
printf("%d ", arr[i]);
return 0;
}
```

Output:

```
2 4 8 12 16 18
```

The below image shows the array created in the above program.



Creating an Array:

The whole process of creating an array can be divided into two primary sub processes i.e.

1. Array Declaration:

Array declaration is the process of specifying the type, name, and size of the array. In C, we have to declare the array like any other variable before using it.

When we declare an array in C, the compiler allocates the memory block of the specified size to the array name.

2. Array Initialization

When the array is declared or allocated memory, the elements of the array contain some garbage value. So, we need to initialize the array to some meaningful values.

- We can skip mentioning the size of the array if declaration and initialisation are done at the same time.
- We can also partially initialize while declaring. In this case, the remaining elements will be assigned the value 0 (or equivalent according to the type).

3. Accessing Array Elements

Array in C provides random access to its elements, which means that we can access any element of the array by providing the position of the element, called the index.

Code:

```
#include <stdio.h>
int main() {
// array declaration and initialization
int arr[5] = {2, 4, 8, 12, 16};
// accessing element at index 2 i.e 3rd element
printf("%d ", arr[2]);
// accessing element at index 4 i.e last element
printf("%d ", arr[4]);
// accessing element at index 0 i.e first element
printf("%d ", arr[0]);
return 0;
}
```

Output

```
8 16 2
```

Update Array Element

We can update the value of array elements at the given index i in a similar way to accessing an element by using the array **square brackets []** and **assignment operator (=)**.

Code:

```
include <stdio.h>
int main() {
   int arr[5] = {2, 4, 8, 12, 16};
  // Update the first value
  // of the array
   arr[0] = 1;
   printf("%d", arr[0]);
   return 0;
}
```

Output

```
1
```

C Array Traversal:

Array Traversal is the process in which we visit every element of the array in a specific order. For C array traversal, we use **loops** to iterate through each element of the array.

Code:

```
#include <stdio.h>
int main() {
   int arr[5] = {2, 4, 8, 12, 16};

   // Print each element of
   // array using loop
   printf("Printing Array Elements\n");
   for(int i = 0; i < 5; i++){
      printf("%d ", arr[i]);
   }
   printf("\n");
   // Printing array element in reverse
   printf("Printing Array Elements in Reverse\n");
   for(int i = 4; i>=0; i--){
      printf("%d ", arr[i]);
   }
   return 0;
}
```

Output

```
Printing Array Elements
2 4 8 12 16
Printing Array Elements in Reverse
16 12 8 4 2
```

Multidimensional Arrays in C - 2D and 3D Arrays

A **multi-dimensional array in C** can be defined as an array that has more than one dimension. Having more than one dimension means that it can grow in multiple directions. Some popular multidimensional arrays include 2D arrays which grows in two dimensions, and 3D arrays which grows in three dimensions.

```
#include <stdio.h>
int main() {
        //Declare and initialize a 2×2 integer array.
        //arr[0][0] = 10, arr[0][1] = 20,
        //arr[1][0] = 30, arr[1][1] = 40
    int arr[2][2] = { {10, 20}, {30, 40} };
    printf("2D Array Elements:\n");
    for (int i = 0; i < 2; i++) {
        for (int j = 0; j < 2; j++) {
            printf("%d ", arr[i][j]);
        }
        printf("\n");
    }
    return 0;
}
```

Syntax

The general form of declaring N-dimensional arrays is shown below:

```
type arrName[size1][size2]....[sizeN];
```

- type: Type of data to be stored in the array.
- arrName: Name assigned to the array.
- size1, size2,..., sizeN: Size of each dimension.

Examples:

```
// Two-dimensional array
int two_d[10][20]
// Three-dimensional array:
int three_d[10][20][30];
```

Problems on Leetcode on Operators:

1.Binary Search
2. Linear/Sequential Search
3. Remove Element
4. remove-duplicates-from-sorted-array
5. Plus One
6. Single Number
7. Contains Duplicate
8. Rotate Array

9. Intersection of two arrays
10. Search in rotated sorted array
11. Search in rotated sorted array II