# Arrays in C

When we work with a large number of data values we need that any number of different variables. As the number of variables increases, the complexity of the program also increases and so the programmers get confused with the variable names. There may be situations where we need to work with a large number of similar data values. To make this work easier, C programming language provides a concept called "Array".

**An array is a special type of variable used to store multiple values of same data type at a time.**

An array can also be defined as follows...

**An array is a collection of similar data items stored in continuous memory locations with single name.**

## Declaration of an Array

We use the following general syntax to create an array...

```
datatype arrayName [ size ] ;
```

Syntax for creating an array with size and initial values

```
datatype arrayName [ size ] = {value1, value2, ...} ;
```

Syntax for creating an array without size and with initial values

```
datatype arrayName [ ] = {value1, value2, ...} ;
```

In the above syntax, the **datatype** specifies the type of values we store in that array and **size** specifies the maximum number of values that can be stored in that array.

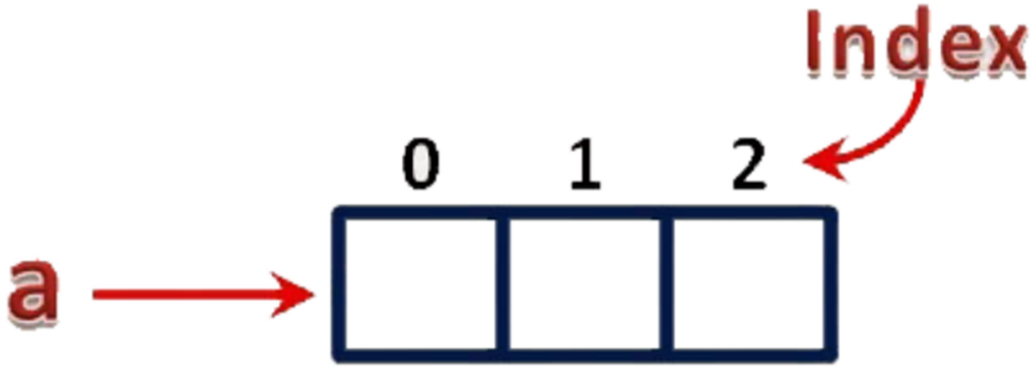
## Example Code

int a [3] ;

Here, the compiler allocates 6 bytes of contiguous memory locations with a single name 'a' and tells the compiler to store three different integer values (each in 2 bytes of memory) into that 6 bytes of memory.

For the above declaration, the memory is organized as follows...


This reference number is called "Index" or "subscript" or "indices". Index values for the above example are as follows...

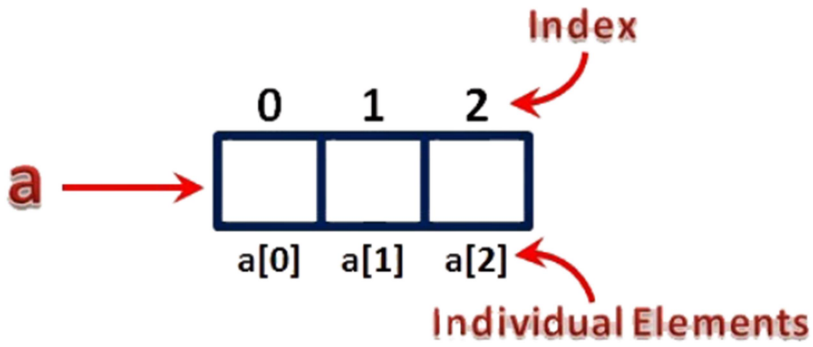


## Accessing Individual Elements of an Array

The individual elements of an array are identified using the combination of 'arrayName' and 'indexValue'. We use the following general syntax to access individual elements of an array...

```
arrayName [ indexValue ] ;
```

For the above example the individual elements can be denoted as follows...

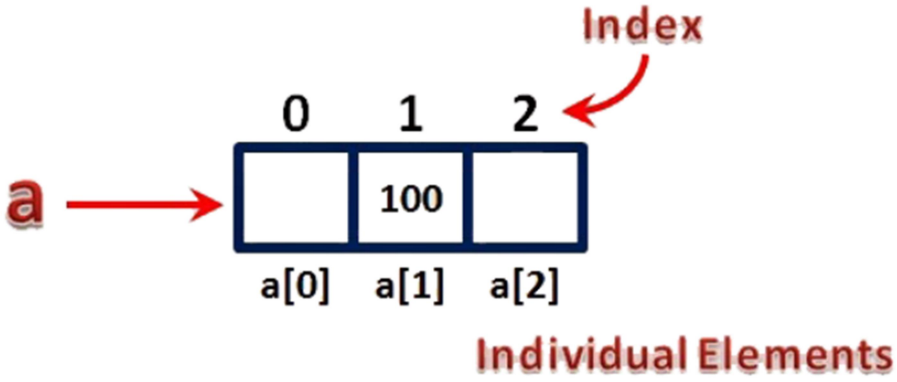


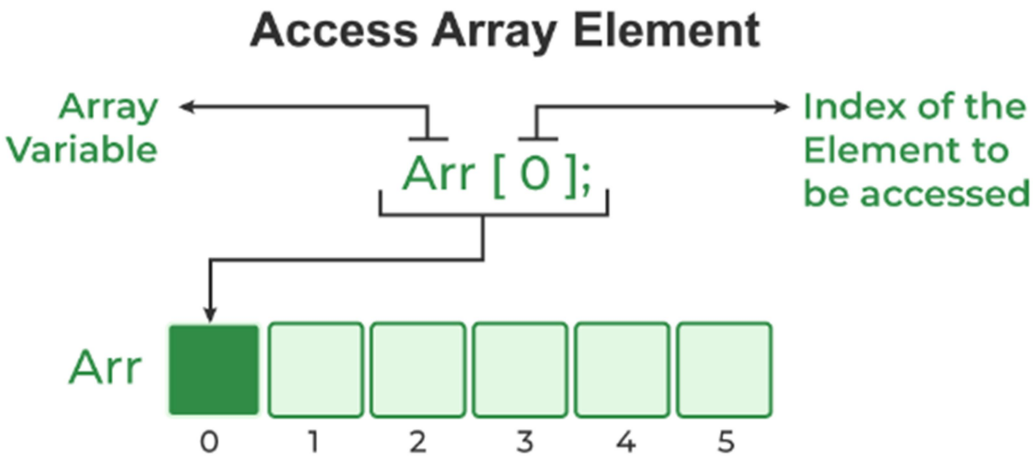

For example, if we want to assign a value to the second memory location of above array 'a', we use the following statement...

### Example Code

```
a [1] = 100 ;
```

The result of the above assignment statement is as follows...

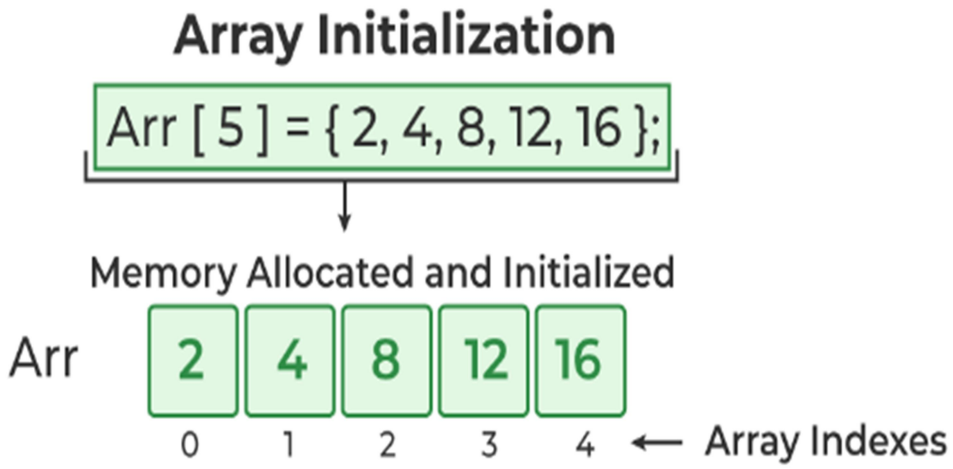





### 1. Array Initialization with Declaration

In this method, we initialize the array along with its declaration. We use an initializer list to initialize multiple elements of the array. An initializer list is the list of values enclosed within braces **{ }** separated b a comma.

```
data_type array_name [size] = {value1, value2, ... valueN};
```



## Array Initialization with Declaration without Size

If we initialize an array using an initializer list, we can skip declaring the size of the array as the compiler can automatically deduce the size of the array in these cases. The size of the array in these cases is equal to the number of elements present in the initializer list as the compiler can automatically deduce the size of the array.

data_type array_name[] = {1,2,3,4,5};

The size of the above arrays is 5 which is automatically deduced by the compiler.

## Array Initialization after Declaration (Using Loops)

We initialize the array after the declaration by assigning the initial value to each element individually. We can use for loop, while loop, or do-while loop to assign the value to each element of the array.

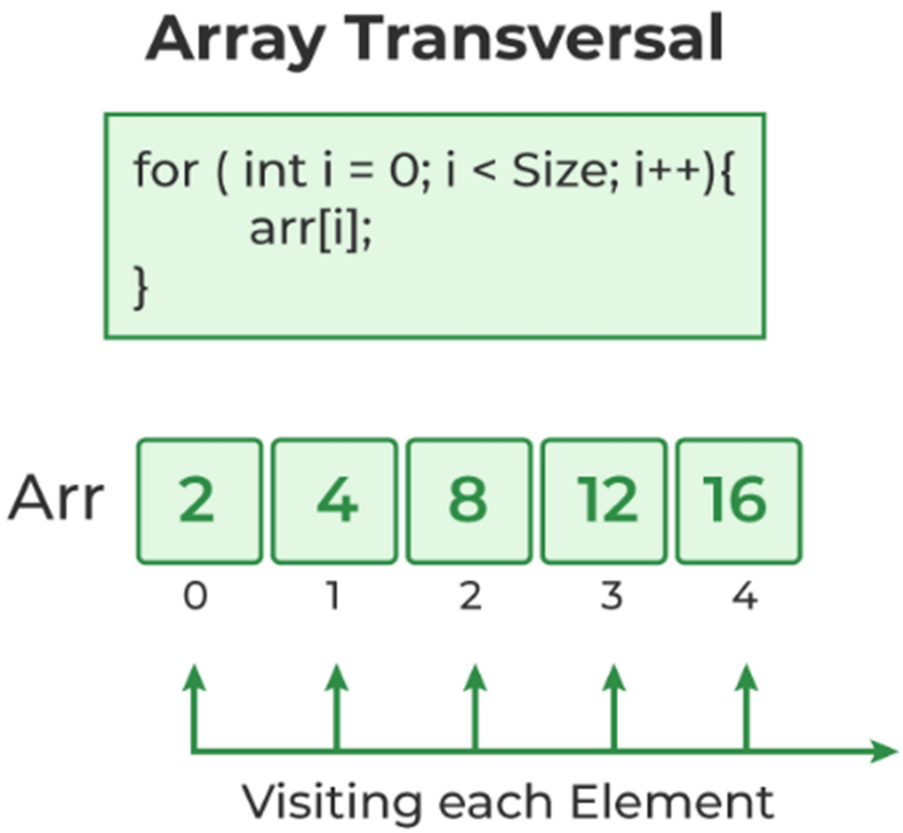
```
for (int i = 0; i < N; i++) {
    array_name[i] = valuei;
}
```

## C Array Traversal

Traversal is the process in which we visit every element of the data structure. For C array traversal, we use loops to iterate through each element of the array.

**Array Traversal using for Loop**

```
for (int i = 0; i < N; i++) {
    array_name[i];
}
```




## Types of Arrays in C

In c programming language, arrays are classified into **two types**. They are as follows...

1. **Single Dimensional Array / One Dimensional Array**
2. **Multi Dimensional Array**

## Single Dimensional Array

In c programming language, single dimensional arrays are used to store list of values of same datatype. In other words, single dimensional arrays are used to store a row of values. In single dimensional array, data is stored in linear form. Single dimensional arrays are also called as **one-dimensional arrays**, **Linear Arrays** or simply **1-D Arrays**.

## Declaration of Single Dimensional Array

We use the following general syntax for declaring a single dimensional array...

```
datatype arrayName [ size ] ;
```

Example Code

```
int rollNumbers [60] ;
```

The above declaration of single dimensional array reserves 60 continuous memory locations of 2 bytes each with the name **rollNumbers** and tells the compiler to allow only integer values into those memory locations.

## Initialization of Single Dimensional Array

We use the following general syntax for declaring and initializing a single dimensional array with size and initial values.

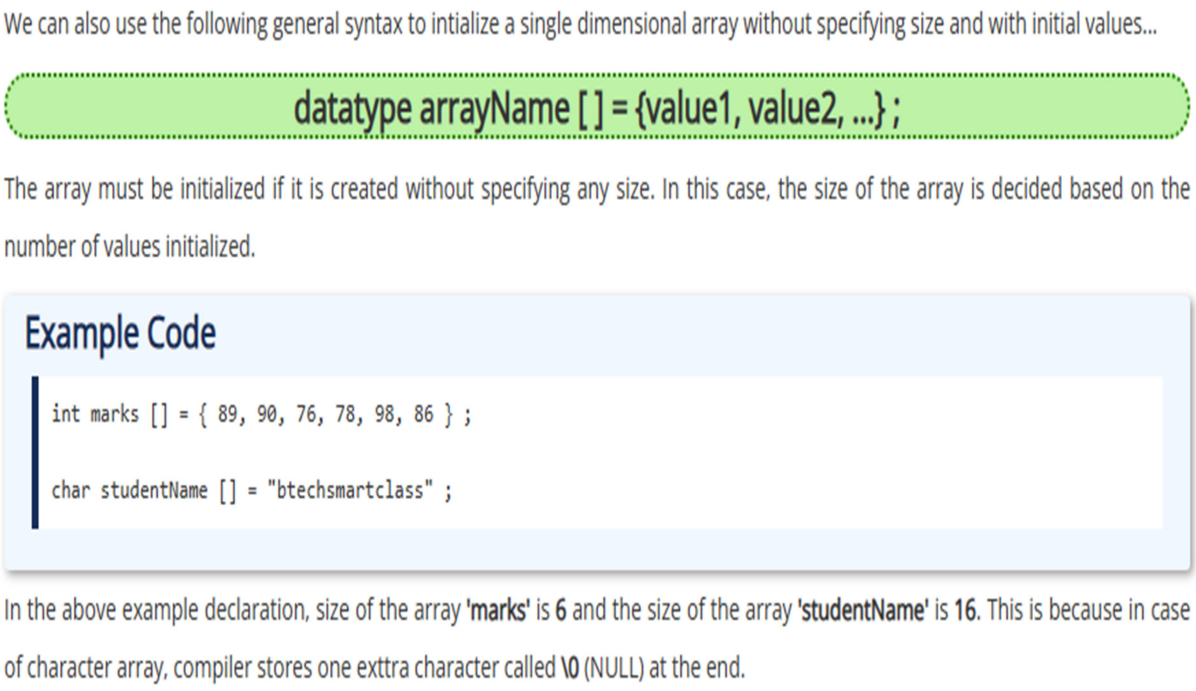
```
datatype arrayName [ size ] = {value1, value2, ...} ;
```

Example Code

```
int marks [6] = { 89, 90, 76, 78, 98, 86 } ;
```

The above declaration of single dimensional array reserves 6 contiguous memory locations of 2 bytes each with the name **marks** and initializes

with value 89 in first memory location, 90 in second memory location, 76 in third memory location, 78 in fourth memory location, 98 in fifth memory location and 86 in sixth memory location.



## Example of 1D Array in C

C

```c
// C Program to illustrate the use of 1D array
#include <stdio.h>

int main()
{

    // 1d array declaration
    int arr[5];

    // 1d array initialization using for loop
    for (int i = 0; i < 5; i++) {
        arr[i] = i * i - 2 * i + 1;
    }

    printf("Elements of Array: ");
    // printing 1d array by traversing using for loop
    for (int i = 0; i < 5; i++) {
        printf("%d ", arr[i]);
    }

    return 0;
}
```

Output

```
Elements of Array: 1 0 1 4 9
```

### Array of Characters (Strings)

In C, we store the words, i.e., a sequence of characters in the form of an array of characters terminated by a NULL character. These are called strings in C language.

C

```c
// C Program to illustrate strings
#include <stdio.h>

int main()
{

    // creating array of character
    char arr[6] = { 'G', 'e', 'e', 'k', 's', '\0' };

    // printing string
    int i = 0;
    while (arr[i]) {
        printf("%c", arr[i++]);
    }
    return 0;
}
```

**Output**

```
Geeks
```

## Multi Dimensional Array:-

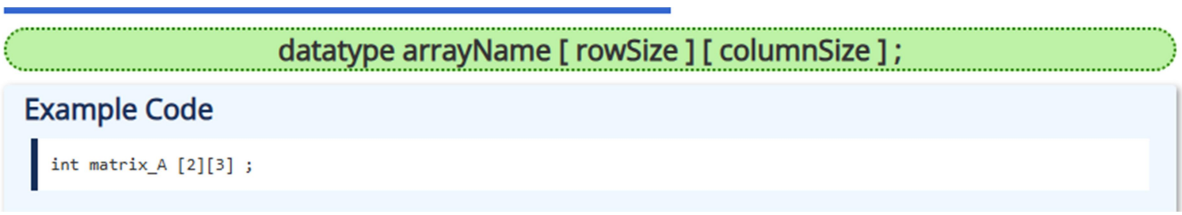
An array of arrays is called as multi dimensional array. In simple words, an array created with more than one dimension (size) is called as multi dimensional array.

Multi dimensional array can be of **two dimensional array** or Most popular and commonly used multi dimensional array is **two dimensional array**.

The 2-D arrays are used to store data in the form of table. We also use 2-D arrays to create mathematical **matrices**.

## Declaration of Two Dimensional Array

We use the following general syntax for declaring a two dimensional array...




The above declaration of two dimensional array reserves 6 continuous memory locations of 2 bytes each in the form of **2 rows** and **3 columns**.

## Two-Dimensional Array in C

A Two-Dimensional array or 2D array in C is an array that has exactly two dimensions. They can be visualized in the form of rows and columns organized in a two-dimensional plane.

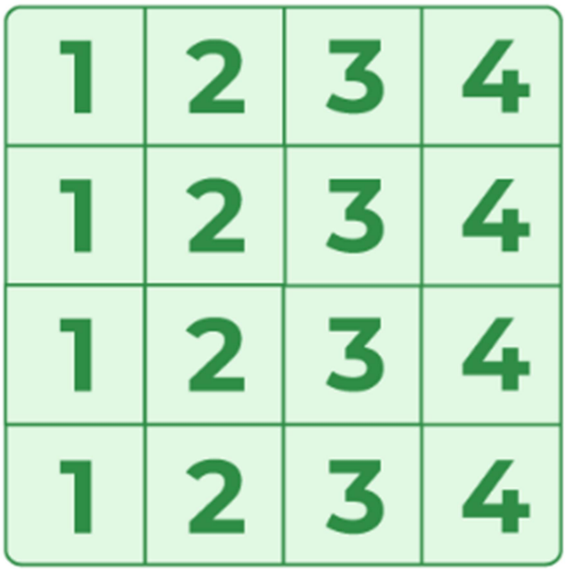
## Syntax of 2D Array in C

array_name[size1] [size2];

Here,

- **size1:** Size of the first dimension.
- **size2:** Size of the second dimension.

# 2D Array



Example of 2D Array in C

C

```c
// C Program to illustrate 2d array
#include <stdio.h>

int main()
{

    // declaring and initializing 2d array
    int arr[2][3] = { 10, 20, 30, 40, 50, 60 };

  printf("2D Array:\n");
    // printing 2d array
    for (int i = 0; i < 2; i++) {
        for (int j = 0; j < 3; j++) {
            printf("%d ",arr[i][j]);
        }
        printf("\n");
    }

    return 0;
}
```

Output

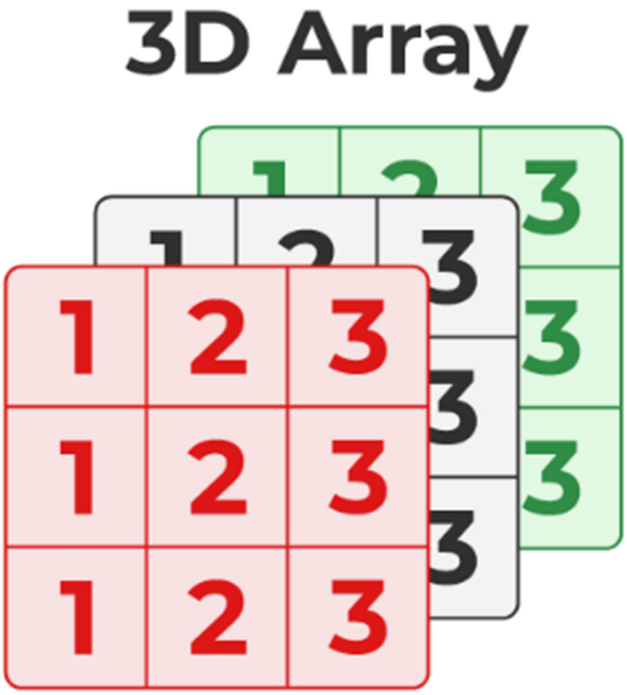
```
2D Array:
10 20 30
40 50 60
```

## B. Three-Dimensional Array in C

Another popular form of a multi-dimensional array is Three Dimensional Array or 3D Array. A 3D array has exactly three dimensions. It can be visualized as a collection of 2D arrays stacked on top of each other to create the third dimension.

### Syntax of 3D Array in C

array_name [size1] [size2] [size3];


3D Array
1 2 3
1 2 3
1 2 3
1 2 3
1 2 3

### Example of 3D Array

C

```c
// C Program to illustrate the 3d array
#include <stdio.h>

int main()
{

    // 3D array declaration
    int arr[2][2][2] = { 10, 20, 30, 40, 50, 60 };

    // printing elements
    for (int i = 0; i < 2; i++) {
        for (int j = 0; j < 2; j++) {
            for (int k = 0; k < 2; k++) {
                printf("%d ", arr[i][j][k]);
            }
            printf("\n");
        }
        printf("\n \n");
    }
    return 0;
}
```

utput

```
10 20
30 40


50 60
0 0
```

## Properties of Arrays in C

It is very important to understand the properties of the C array so that we can avoid bugs while using it. The following are the main properties of an array in C:

**1. Fixed Size**

The array in C is a fixed-size collection of elements. The size of the array must be known at the compile time and it cannot be changed once it is declared.

**2. Homogeneous Collection**

We can only store one type of element in an array. There is no restriction on the number of elements but the type of all of these elements must be the same.

**3. Indexing in Array**

The array index always starts with 0 in C language. It means that the index of the first element of the array will be 0 and the last element will be N – 1.

**4. Dimensions of an Array**

A dimension of an array is the number of indexes required to refer to an element in the array. It is the number of directions in which you can grow the array size.

**5. Contiguous Storage**

All the elements in the array are stored continuously one after another in the memory. It is one of the defining properties of the array in C which is also the reason why random access is possible in the array.

**6. Random Access**

The array in C provides random access to its element i.e we can get to a random element at any index of the array in constant time complexity just by using its index number.

**7. No Index Out of Bounds Checking**

There is no index out-of-bounds checking in C/C++, for example, the following program compiles fine but may produce unexpected output when run.

## Applications of Arrays in C

In c programming language, arrays are used in wide range of applications. Few of them are as follows...

- **Arrays are used to Store List of values**

In c programming language, single dimensional arrays are used to store list of values of same datatype. In other words, single dimensional arrays are used to store a row of values. In single dimensional array data is stored in linear form.

- **Arrays are used to Perform Matrix Operations**

We use two dimensional arrays to create matrix. We can perform various operations on matrices using two dimensional arrays.

- **Arrays are used to implement Search Algorithms**

We use single dimensional arrays to implement search algorihtms like ...

1. **Linear Search**
2. **Binary Search**

- **Arrays are used to implement Sorting Algorithms**

We use single dimensional arrays to implement sorting algorihtms like ..

1. **Insertion Sort**

2. Bubble Sort
3. **Selection Sort**
4. **Quick Sort**
5. Merge Sort, etc.,

- **Arrays are used to implement Datastructures**

We use single dimensional arrays to implement datastructures like...

1. **Stack Using Arrays**
2. **Queue Using Arrays**

- **Arrays are also used to implement CPU Scheduling Algorithms.**