

Array :- Arrays are defined as collection of similar type of data items stored at contiguous memory locations.

Array is the simplest data structure where each data element can be randomly accessed by using its index number.

Array declaration :-

```
int arr [10] ; char arr [10] ; float arr [5]
```

Program without Array :-

```
#include <stdio.h>
void main ()
{
    int marks-1 = 56 ; marks-2 = 78 , marks-3 = 89 ;
    float avg = (marks-1 + marks-2 + marks-3) / 3 ;
    print (avg) ;
}
```

Program by using Array :-

```
#include <stdio.h>
void main
{
    int marks [3] = { 56 , 78 , 89 } ;
    int i ;
    float avg ;
    for (i = 0 ; i < 3 ; i++)
```

```

    {
        avg = avg + marks[i];
    }
    printf (avg);
}

```

Complexity of Array operations :-

1) Time complexity :-

Algorithm	Average case	worst case
Access	$O(1)$	$O(1)$
search	$O(n)$	$O(n)$
insertion	$O(n)$	$O(n)$
Deletion	$O(n)$	$O(n)$

2) Space complexity :-

In Array space complexity for worst case is $O(n)$

Memory Allocation of the Array :-

Each element in Array represented by indexing
Indexing of array can be defined in three ways:

1. 0 (Zero Based indexing) :-

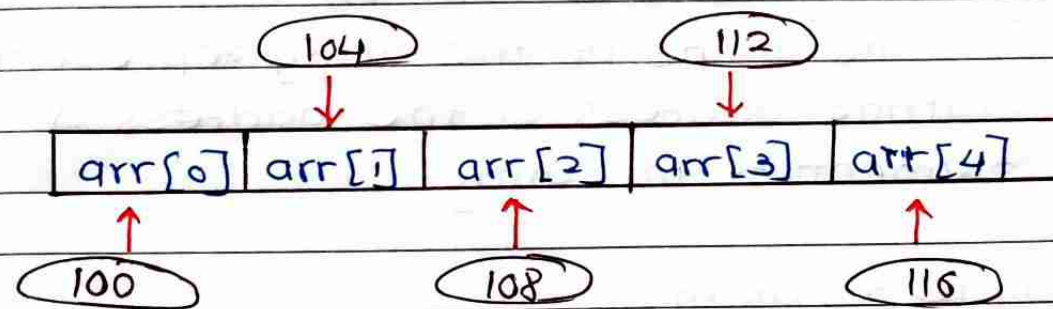
The first element of the array will be $arr[0]$.

2. 1 (one-based indexing) :-

The first element of array will be $\text{arr}[1]$.

3. n (n-based indexing) :-

The first element of array can reside at any random index number.



Base address.

(fig: $\text{int arr}[5]$)

Accessing elements of an Array :-

To access any random element of an array we need the following information:

1. Base address of the array
2. size of an element in bytes.
3. which type of indexing, array follows.

Address of any element of 1D array can be calculated

$$\text{Byte address of element } A[i] = \text{base address} + \text{size} \times (\text{first-index})$$

Example: In an array, $A[-10 \dots +2]$ Base address (BA) = ggg, size of an element = 2 bytes, find location of $A[-i]$.

solution: $L(A[-1]) = 999 + [(-1) - (-10)] \times 2$

$$= 999 + 18$$

$$= 1017$$

\therefore location of $A[-1] = 1017$

Passing array to the function :-

The name of the array represents the starting address or the address of the first element of the array.

Program:

```
#include <stdio.h>
int summation (int []);
void main ()
{
    int arr [5] = {0, 1, 2, 3, 4};
    int sum = summation (arr);
    printf ("%d", sum);
}

int summation (int arr[])
{
    int sum = 0, i;
    for (i = 0; i < 5; i++)
    {
        sum = sum + arr [i];
    }
    return sum;
}
```


2D Array :- 2D array can be defined as an array of arrays. The 2D array is organized as matrices which can be represented as collection of rows and columns.

How to declare 2D Array :-

The syntax for declaration of two dimensional array is as follows :

`int arr [max - rows] [max - columns];`

However, it produces the data structure which looks like following :

	0	1	2	...	n-1
0	$a[0][0]$	$a[0][1]$	$a[0][2]$	$a[0][n-1]$
1	$a[1][0]$	$a[1][1]$	$a[1][2]$	$a[1][n-1]$
2	$a[2][0]$	$a[2][1]$	$a[2][2]$...	$a[2][n-1]$
\vdots	\vdots	\vdots	\vdots	...	\vdots
n-1	$a[n-1][0]$	$a[n-1][1]$	$a[n-1][2]$		$a[n-1][n-1]$

$a[n][n]$

(Fig : $a[n][n]$)

How to access data in 2D-array :-

Due to fact that elements of 2D arrays can be random accessed.

$\text{int } x = a[i][j];$

where i, j are the rows and columns respectively.

Initializing 2D arrays :-

The syntax to declare and initialize the 2D array is given as follows :

$\text{int arr}[2][2] = \{0, 1, 2, 3\};$

number of elements in 2D arrays

= number of rows * number of columns.

Mapping 2D array to 1D array :-

The size of a two dimensional array is equal to the multiplication of number of rows and number of columns present in the array.

A 3×3 two dimensional array is shown:-

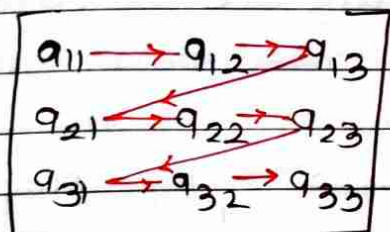
	0	1	2	
0	(0,0)	(0,1)	(0,2)	column index
1	(1,0)	(1,1)	(1,2)	
2	(2,0)	(2,1)	(2,2)	

row index

There are two main techniques of storing 2D array elements into memory.

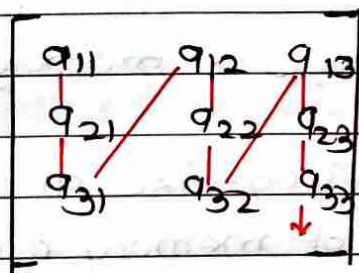
1. Row major ordering :-

In row major ordering, all the rows of 2D array are stored into memory contiguously.



2. Column major ordering :-

According to column major ordering, all the columns of 2D array are stored into the memory contiguously.



Calculating address of random element of a 2D array :-

1) By row major order :-

If array is declared $a[m][n]$ where m is the number of rows while n is number of columns. then address of an element $a[i][j]$ is calculated as,

$$\text{Address}(a[i][j]) = B.A + (i * n + j) * \text{size}$$

$B.A \rightarrow$ Base Address

2) By column major order :-

$$\text{Address}(a[i][j]) = (j * m + i) * \text{size} + B.A.$$