

Types of Array :-

→ There are two types of Array.

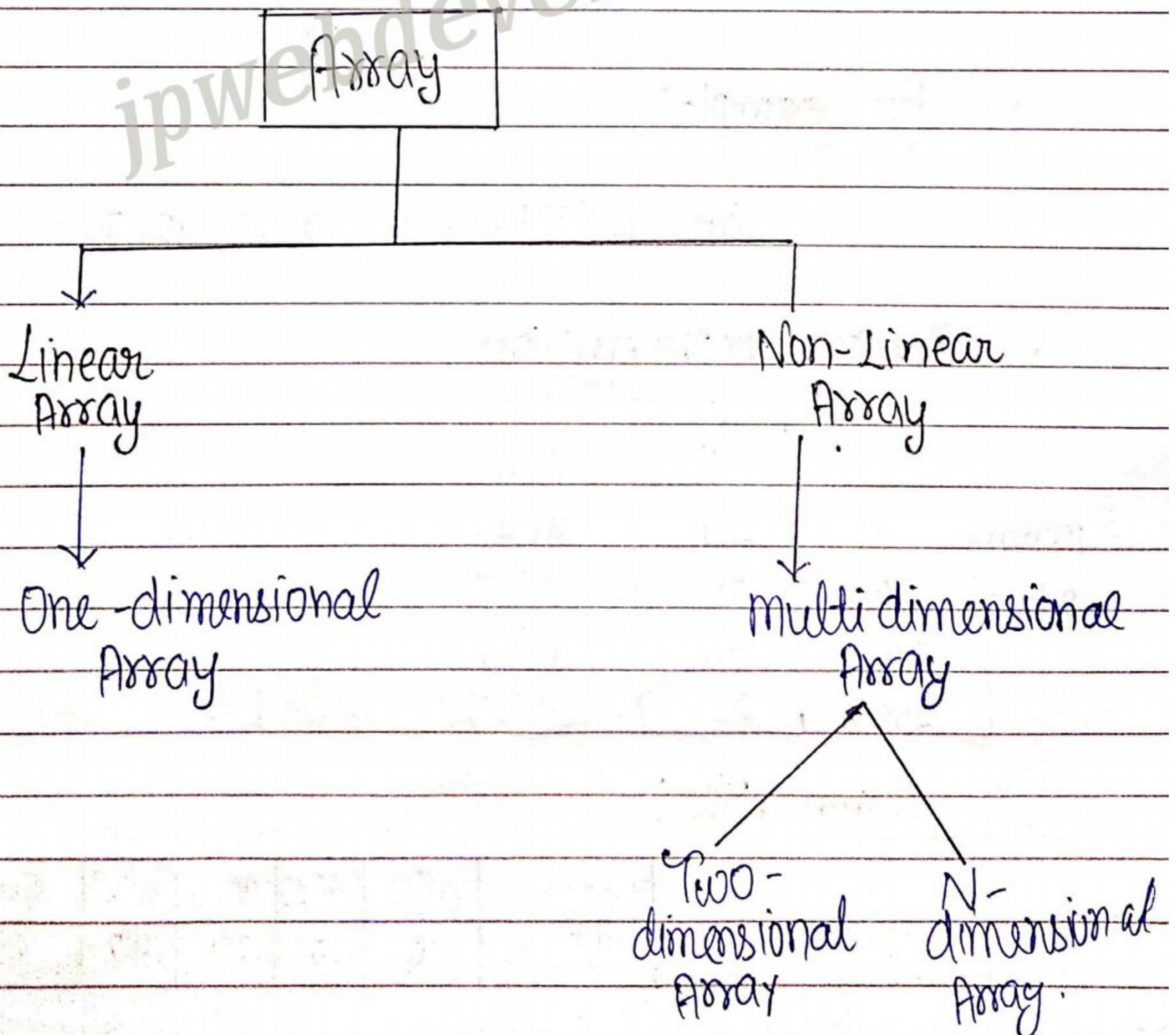
1. Linear Array

→ One Dimensional Array.

2. Non-Linear Array

→ Two Dimensional Array
or

→ Multi Dimensional Array



1. One-dimensional Array:-

- A one-dimensional Array is also known as single dimensional array or 1D array or linear Array.
- One dimensional array has only a single subscript.
- This type of array will be accessed by the subscript ~~of~~ either column wise or row wise.
- for example:-

```
int a[5] = { 10, 20, 30, 40, 50 }
```

Memory representation

Memory Address	2000	10	a[0]
	2002	20	a[1]
	2003	30	a[2]
	2004	40	a[3]
	2005	50	a[4]

Data elements →

index

Column wise:-

Index	a[0]	a[1]	a[2]	a[3]	a[4]
Data elements	10	20	30	40	50

• Array Declaration:-

→ Syntax to declare an array:-

```
type array-name [size];
```

→ "type", is the data type.

→ "array name" name of the array, must be unique name.

→ "Size" must be an integer constant

```
Example:- int arr [5];
```

• Array Initialization:-

→ Initialization means assigning a value at the time of declaration.

→ As we have initialized the variable, in the same way we can initialize array.

→ Syntax to initialize array:-

```
data_type array_name [size] = {value list separated by comma};
```

→ Example:- int marks [5] = {10, 20, 30, 40, 50};

* Array Accessing :-

→ array can access with the help of loop.

→ An array element is accessed by writing the identifier of the array followed by the subscript.

#1 Program to Entering Data into an array from user.

```
#include <stdio.h>
#include <conio.h>
void main()
{
```

```
    int marks [5],
    int i;
    for (i=0; i<5; i++)
    {
```

```
        printf ("Enter marks");
        scanf ("%d", &marks[i]);
```

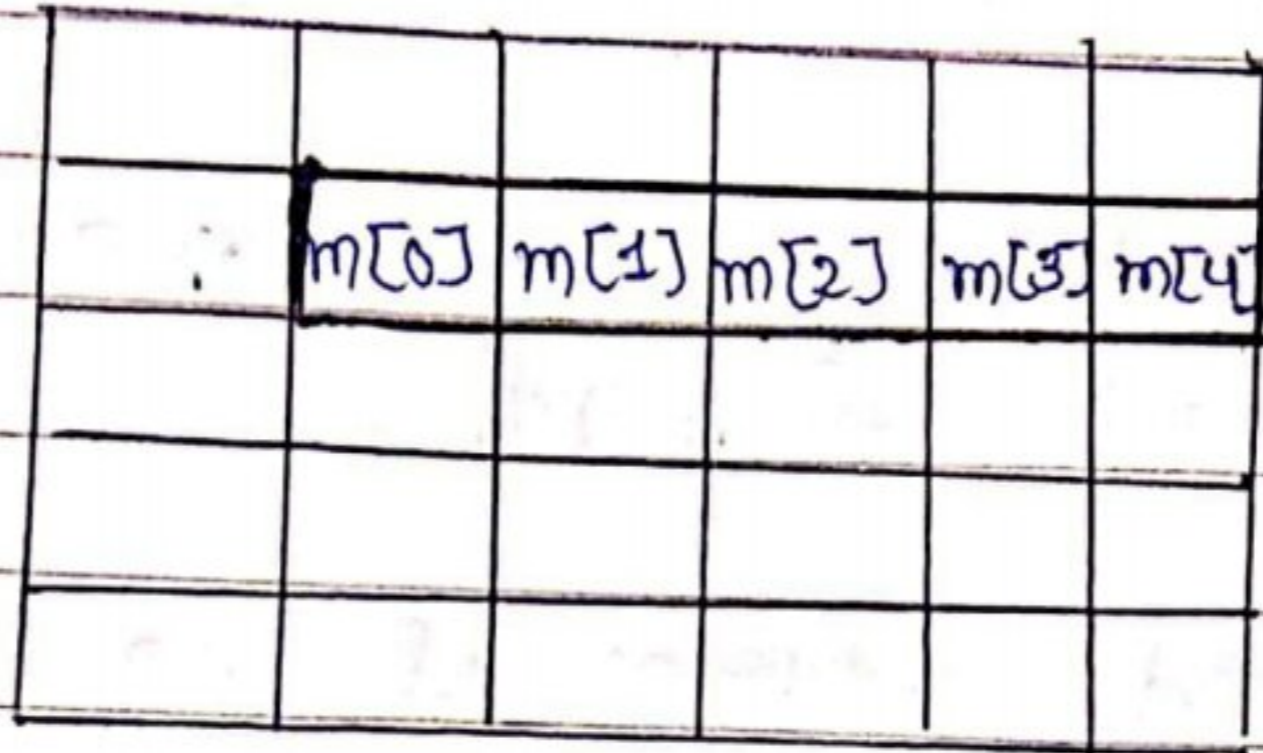
```
    }
    for (i=0; i<5; i++)
    {
```

```
        printf ("Subject in %d Marks is %d",
        i+1, marks[i]);
```

```
    }
    getch();
}
```

* Memory Representation of one-dimensional array:

→ In case of array all the elements are stored in a contiguous memory location.



Here elements are stored in contiguous memory location.

For example

m[0]	m[1]	m[2]	m[3]	m[4]
------	------	------	------	------

2002 2004 2006 2008 2010

← memory address

* Calculate the Address of element in the array:-

Formula:-

$$\text{Location of array (Address of element)} = \text{Base address} + \text{Size} * (\text{index} - \text{lower bound})$$

- _ / _ / _
- Base Address of the array
 - Size of an element in byte
 - type of indexing

- Example Suppose array with L.B = 1
(First Index)
arr[10], Base address = 1020, and
Size of each element = 2

Find address of arr[5] ?

Ans

$$\text{Base (A)} = 1020$$

$$\text{Size} = 2$$

$$\text{first index} = 1$$

$$\text{Add(arr[5])} = 1020 + 2 (5 - 1)$$

$$= 1020 + 2 (4)$$

$$= 1020 + 8$$

$$= 1028$$

$$\text{arr[5]} = 1028$$

* Two Dimensional Array :- (Multi-d Array)

→ An array can have more than one series of element. It is called a Multidimensional Array

→ When the number of dimensions specified is more than one, it is called as Multi-dimensional Array.

→ Multidimensional Arrays includes 2D arrays and 3D arrays.

→ A Two Dimensional Array will be accessed by using the subscript of row and column index

`int a [2][3] = {10, 20, 30, 40, 50, 60};`

↑ ↑
number number
of of
rows columns

→ Array can hold 6 elements ($2 * 3$).

→ Memory Map:

	Col 0	Col 1	Col 2
Row 0	10	20	30
Row 1	40	50	60

memory address	2000	10	a[0][0]
	2002	20	a[0][1]
	2004	30	a[0][2]
	2006	40	a[1][0]
	2008	50	a[1][1]
	2010	60	a[1][2]

Data elements

← Index

* Array Declaration:-

Syntax to declare two

```
type array_name[row_size][col_size];
```

- "type" is the data type, that specify which type of elements will array store.
- "array_name" name of the array.
- "row_size" must be an integer constant.
- "col_size" must be an integer constant.

Example:- `int marks[3][4];`

↑ data type ↑ array name ↑ No. of rows ↑ No. of cols

* Array Initialization:-

The Syntax to initialize an array as follows:-

1. datatype arrayname [row_size] [col_size] = { value list
separated by
Comma };
2. datatype array_name [] [col_size] = { value list by
separated by comma };

Example: `int [2] [2] = { 0, 1, 2, 3 };`
= = =

`int [2] [2] = { 0, 1, 2, 3 };`
Diagram illustrating the initialization of a 2D array. The array is defined as `int [2] [2]`. The values `0, 1, 2, 3` are listed in curly braces. Arrows indicate that the first two values (`0, 1`) are assigned to the first row, and the next two values (`2, 3`) are assigned to the second row.

Example of Two-Dimensional Array (Insertion of 2D Array)

```
void main()
{
    int a [3] [4], i, j;
    for (i=0; i<3; i++)
    {
        for (j=0; j<4; j++)
        {
            printf ("Enter number");
            scanf ("%d", &a[i][j]);
        }
    }

    for (i=0; i<3; i++)
    {
        for (j=0; j<4; j++)
        {
            printf ("%d\t", a[i][j]);
        }
        printf ("\n");
    }
    getch();
}
```


- Base (A), is the base address of the array.
- W, scale factor by which cells are increased.
- No. → total No. of Columns in Array.
- i → row location, j → column location.
- (-1) to be performed only if array location will not start from 0,0.

• for example

Row 0	5001	5003	5005
Row 1	5007	5009	5011
Row 2	5013	5015	5017
Row 3	5019	5021	5023

Qus To find the address of the cell [2,2] if cell locations starts from [1,1].

$$\begin{aligned}
 (A[2,2]) &= 5001 + 2 [3(2-1) + (2-1)] \\
 &= 5001 + 2 [3(1) + 1] \\
 &= 5001 + 2 [4] \\
 &= 5001 + 8 \\
 &= 5009
 \end{aligned}$$

	Col 1	Col 2	Col 3
Row 1	5001	5003	5005
Row 2	5007	5009	5011
Row 3	5013	5015	5017
Row 4	5019	5021	5023

2. Column Major Array :-

• Column Major ordering is the other function frequently used to compute the address of an array element.

• In Column major array elements are stored column by column.

• For example:-

int A [4] [3];

• formula to find LOC[A].

$$\text{LOC}[A[i, j]] = \text{Base}(A) + w [M \underset{\substack{\downarrow \\ \text{no.} \\ \text{of rows}}}{(j-1)} + \underset{\substack{\uparrow \\ \text{row} \\ \text{location}}}{(i-1)}]$$

Col. location

→ Base(A), is the base address of the array.

→ w, is the scale factor.

→ M, is total no. of rows

→ I → is the row location, j is the Col. location

→ minus 1, is to be performed only if array location will not to start from (0,0)

Qus:- To find the address of the cell [3,2], if cell locations start from [1,1].

Base address = 5001

Ans

$$\begin{aligned} \text{Loc}(A[3,2]) &= 5001 + 2[4(2-1) + (3-1)] \\ &= 5001 + 2[4(1) + 2] \\ &= 5001 + 2[4+2] \\ &= 5001 + 2(6) \\ &= 5001 + 12 \\ &= 5013 \end{aligned}$$

	Col 1	Col 2	Col 3
Row 1	5001	5009	5017
Row 2	5003	5011	5019
Row 3	5005	5013	5021
Row 4	5007	5015	5022