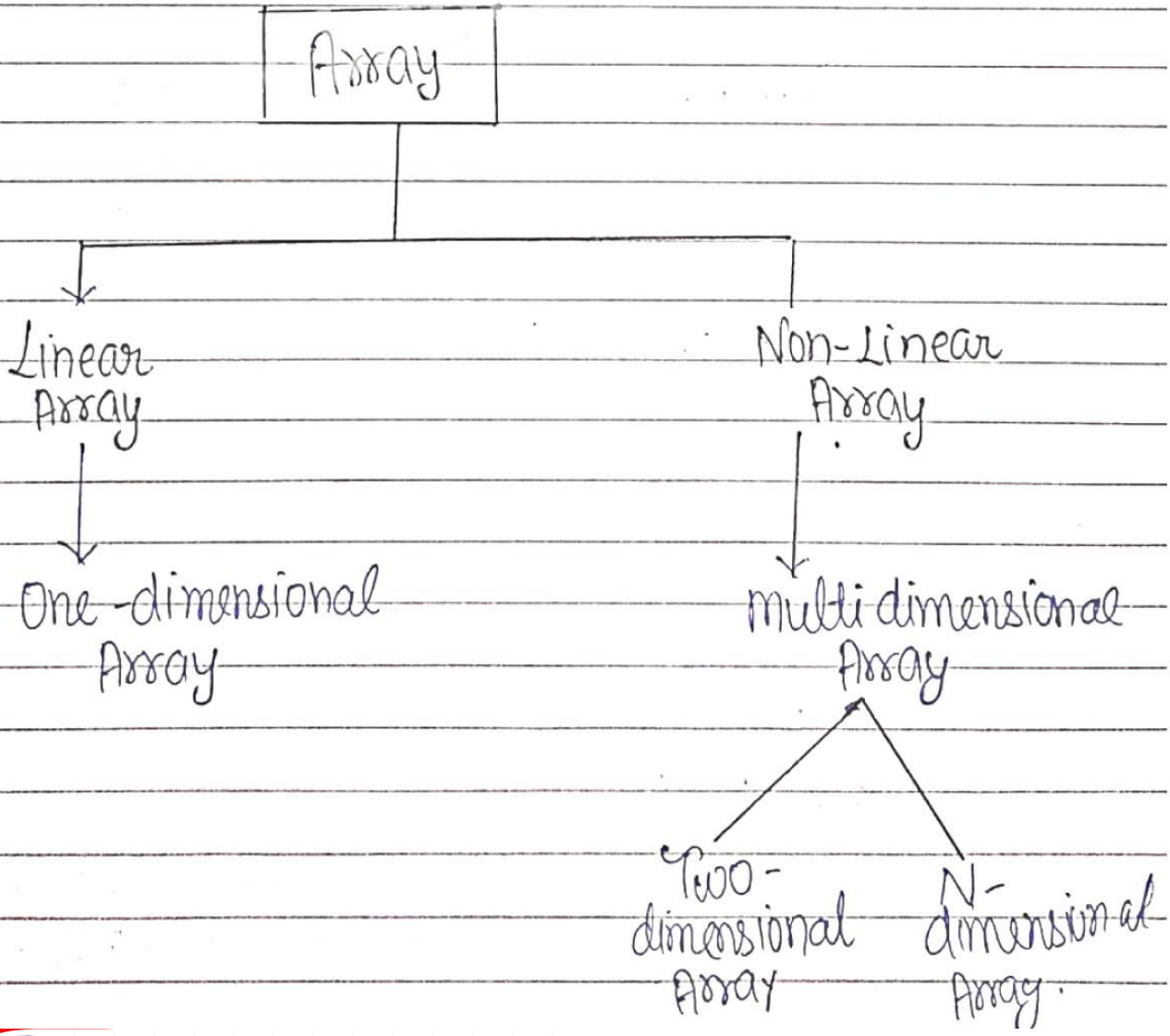
__/__/____

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	Data elements	index
2000	10	a[0]
2002	20	a[1]
2003	30	a[2]
2004	40	a[3]
2005	50	a[4]

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:-

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

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

→ Syntax to intialize 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.

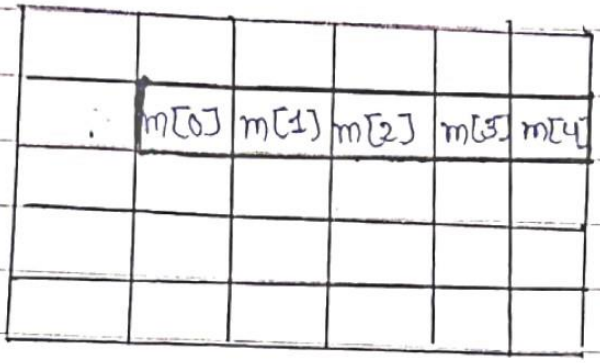
Program to Entering Data into an array from user.

```
#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:-

Location of array (Address of element) = Base address + Size * (i - first index)

first index ↑ 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

Base (A) = 1020
size = 2
first index = 1

Add(arr[5]) = 1020 + 2 (5 - 1)
= 1020 + 2 (4)
= 1020 + 8
= 1028

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

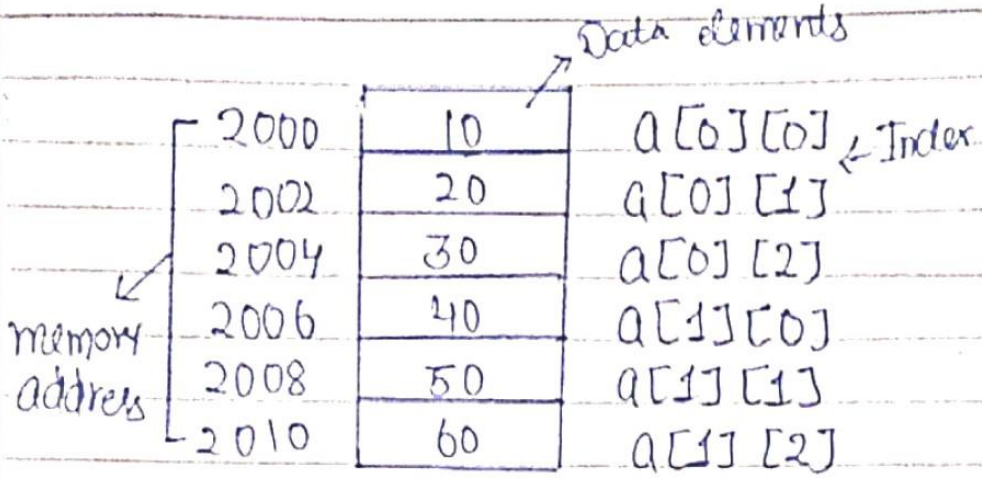
```
int a[2][3] = {10,20,30,40,50,60};
```

number of rows — number of 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



* Array Declaration:-

Syntax to declare 2D array

```
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 :-

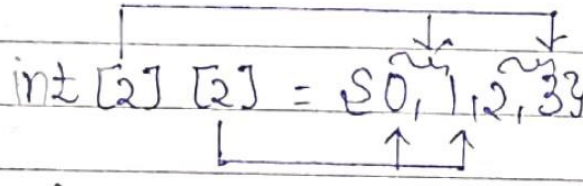
The Syntax to initialized an array as follows :-

1. datatype arrayname[row_size][col_size] = { value list seperated by comma };

or

2. datatype array_name[][colsize] = { value list by separated by comma };

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



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();
```

__/__/____

* Memory Representation of 2-dimensional Array:-

→ Representation of 2-d array in memory is row major and column major.

→ A 2D Array has a type such as int[], with two pairs of square brackets.

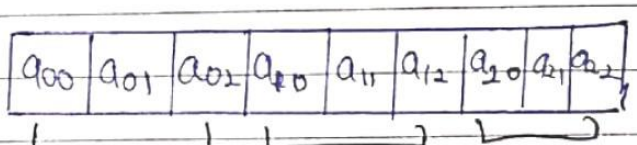
```
int s[4][2];
    ↑  ↑
    no. of rows   no. of cols
```

1. Row-Major Order:-

→ Elements are stored Row wise.
→ Rows are listed on the basis of columns.

For example:

$$a = \begin{bmatrix} a_{00} & a_{01} & a_{02} \\ a_{10} & a_{11} & a_{12} \\ a_{20} & a_{22} & a_{22} \end{bmatrix} \rightarrow$$



Address of element (A[I,J])

$$= Base(A) + W(N(I-1) + (J-1))$$

- Base(A): Base address
- W: size of element
- N: no. of cols
- (I-1): lower bound of row (← row location)
- (J-1): lower bound of column (← column location)

__/__/____

→ 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].

$$(A[2,2]) = 5001 + 2[3(2-1) + (2-1)]$$
$$= 5001 + 2[3(1) + 1]$$
$$= 5001 + 2[4]$$
$$= 5001 + 8$$
$$= 5009$$

	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 orther 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].

$$LOC[A[i,j]] = Base(A) + w[M(j-1) + (i-1)]$$

(Col. location; M → no. of rows; (i-1) → row 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 $Loc(A[3,2]) = 5001 + 2[4(2-1)+(3-1)]$

$= 5001 + 2[4(1) + 2]$

$= 5001 + 2[4+2]$

$= 5001 + 2(6)$

$= 5001 + 12$

$= 5013$

	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