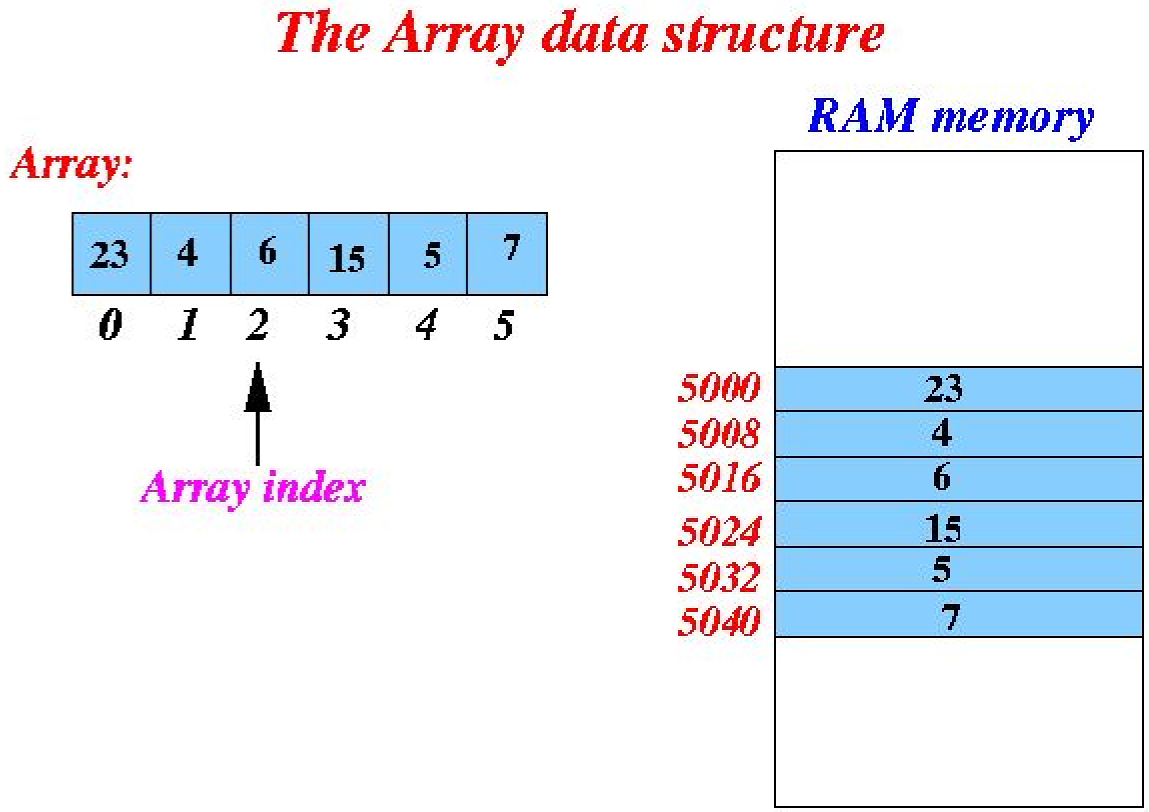
Array

Main Points :

- What Is Array ?
- Advantages Of Array
- Disadvantages Of Array
- Time Complexity
- Space Complexity
- 2D Array
- Resources

What Is Array ?

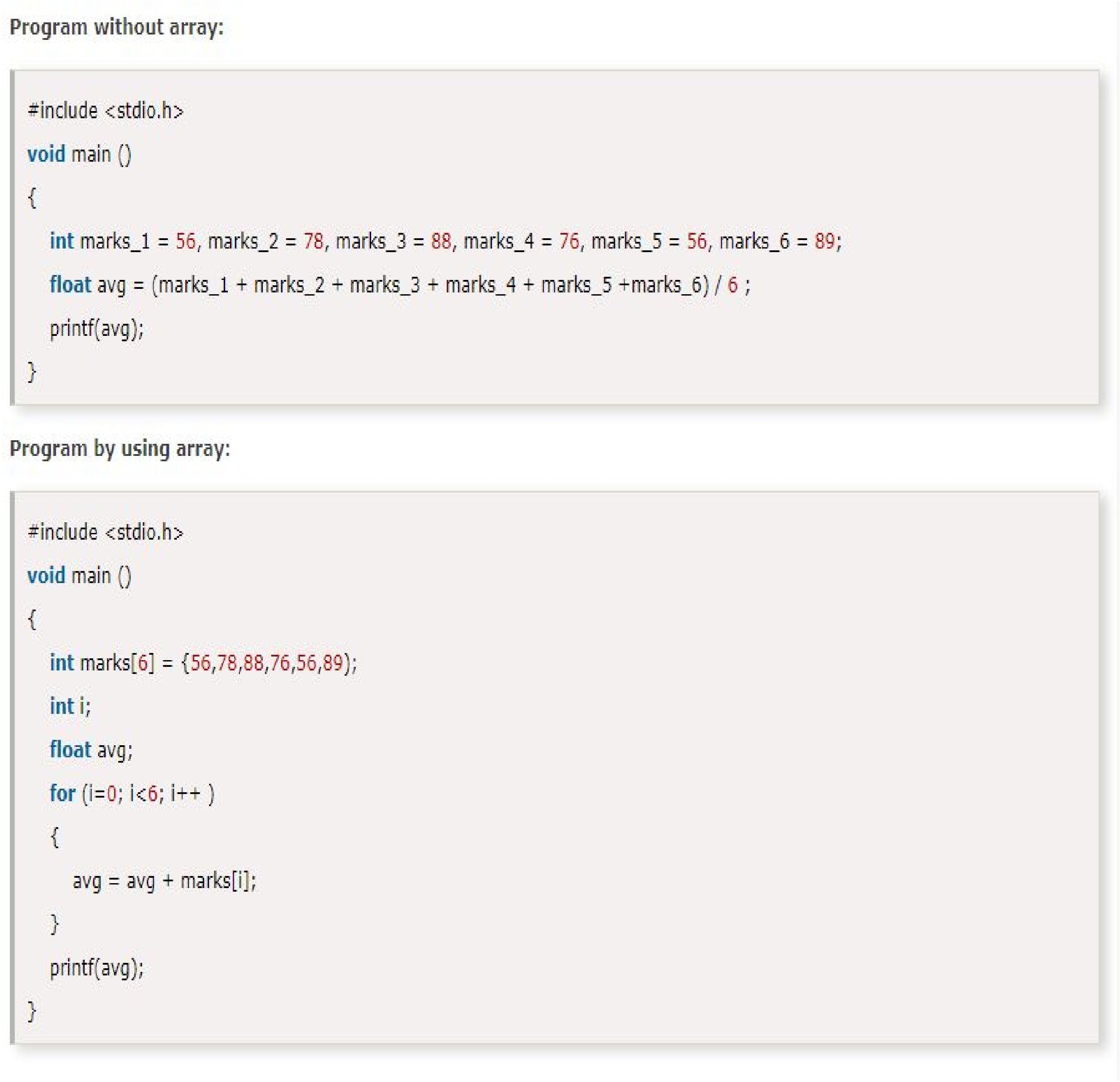
- collection of similar type of data items stored at contiguous memory locations.



Advantages Of Array :

1- Single Name , Multi Values

Instead Of Define Large Numbers Of Variables To Store Similer Data Type Items .. Array Has Single Name That Can Store Multi Values in it.



Program without array:

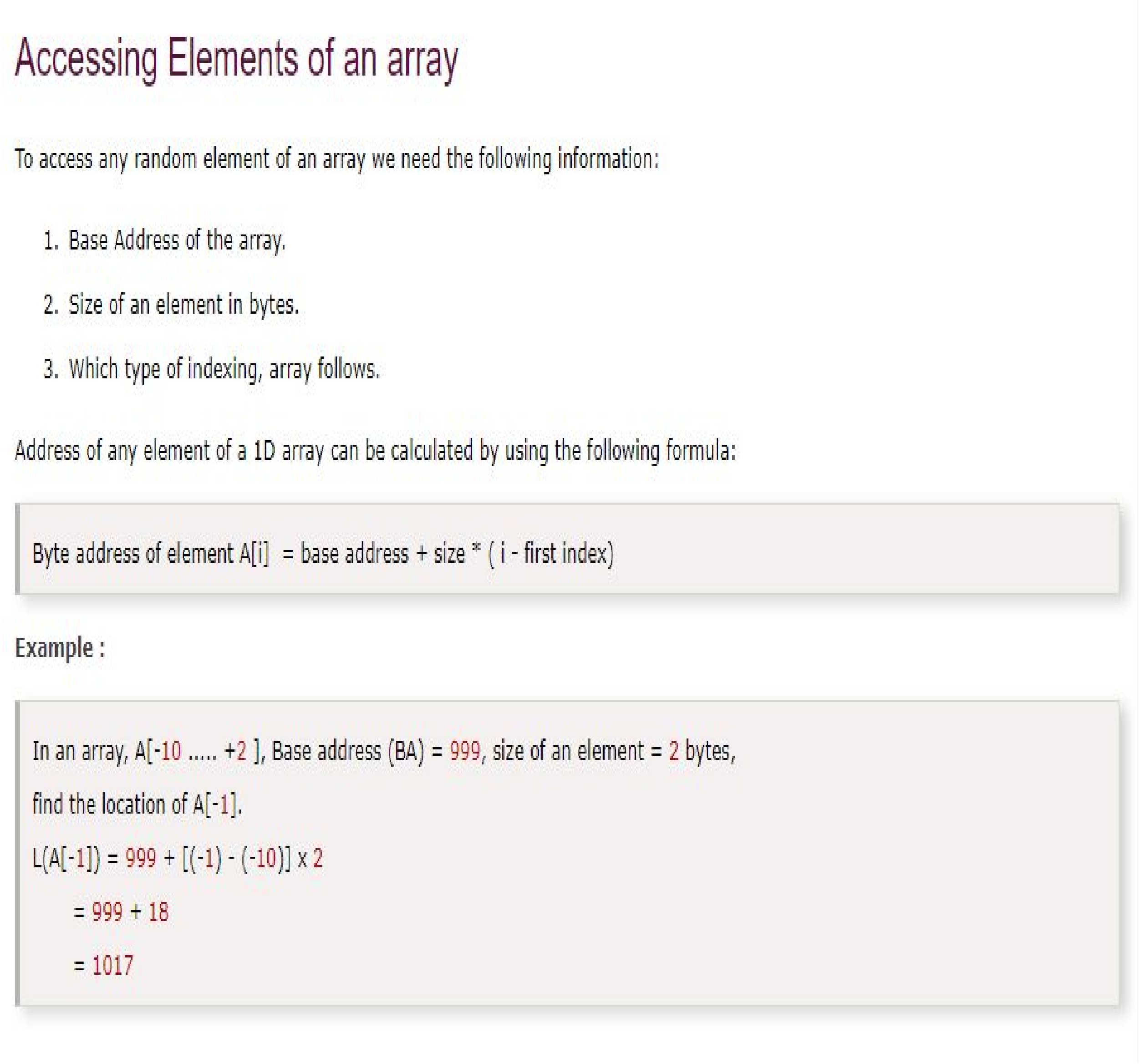
```
#include <stdio.h>
void main ()
{
   int marks_1 = 56, marks_2 = 78, marks_3 = 88, marks_4 = 76, marks_5 = 56, marks_6 = 89;
   float avg = (marks_1 + marks_2 + marks_3 + marks_4 + marks_5 +marks_6) / 6 ;
   printf(avg);
}
```

Program by using array:

```
#include <stdio.h>
void main ()
{
   int marks[6] = {56,78,88,76,56,89};
   int i;
   float avg;
   for (i=0; i<6; i++ )
   {
      avg = avg + marks[i];
   }
   printf(avg);
}
```

2- Direct Access Using Index

Any element in any position in the array can be directly accessed using the index in just one step Because We Have Equation That Calculate Memory Address Of That Element.
Access Element Mean Read Or Write Value On It.
Access Element in Array --> Big(o) = 1



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 a 1D array can be calculated by using the following formula:

> Byte address of element A[i] = base address + size * (i - first index)

Example :

> In an array, A[-10 ..... +2], Base address (BA) = 999, size of an element = 2 bytes,
>
> find the location of A[-1].
>
> L(A[-1]) = 999 + [(-1) - (-10)] x 2
>
> = 999 + 18
>
> = 1017

3- Array is a foundation of other data structures.

For example other data structures such as LinkedList, Stack, Queue etc. are implemented using array.

Disadvantages Of Array :

1- Fixed Size Or Static Structure :

1.1 We Need To Know The Size Of Array Before We Declare It

1.2 Empty Or Unused Elements Caused Waste In Memory

If We Need Array Of 50 Elements And We Declare Array Of 100 Elements , So We Waste a lot Of Memory Size

1.3 ReCreate New Array To Extend Old Array Size

If We Need Array Of 50 Elements And Declare Array Of 50 Elements , But In The RunTime We Need To Extend The Size Of Array To 80 Elements , We Can Not Do That , And We Need To Declare New Array With Size Of 80 And Copy Old Elements In The New Array

1.4 Insert New Element Is Very Slow : O(N)

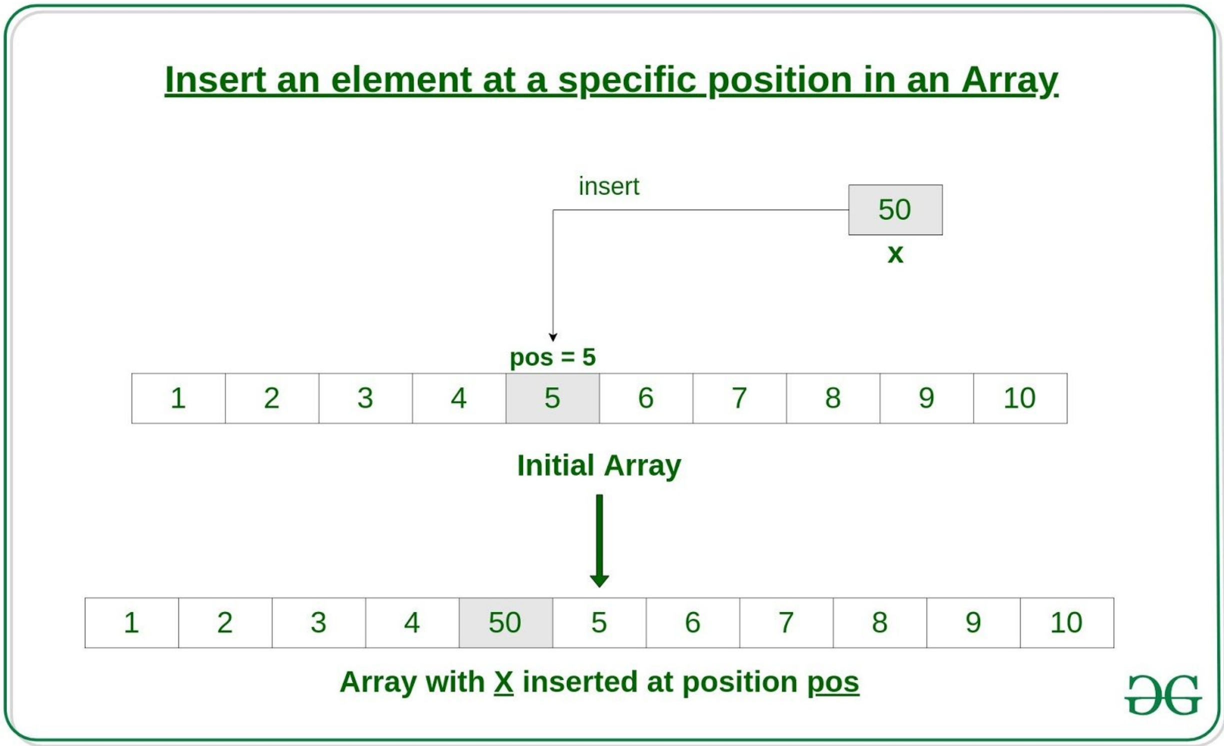
Actually We Can not Insert New Element Directly In Our Array Because it Has Static Structure.But We

1- Create New Array That Has Higher Size.

2- Copy Elements From Old Array To New Array Until We Reached To The Position Required.

3- Insert New Element At this Position.

4- Copy And Shift Remaining Items.



Insert an element at a specific position in an Array

Array with X inserted at position pos

1.5 Delete Element Is Very Slow : O(N)

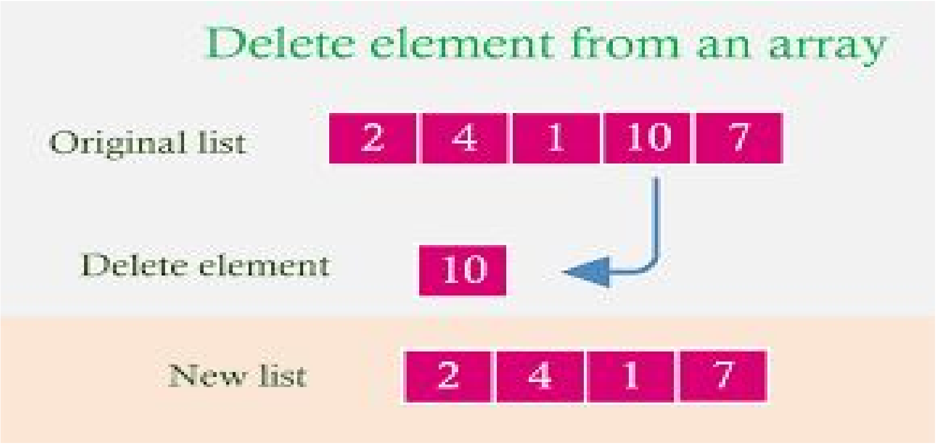
Actually We Can not Remove Element Directly From Our Array Because it Has Static Structure.But We

1- Create New Array With Lower Size.

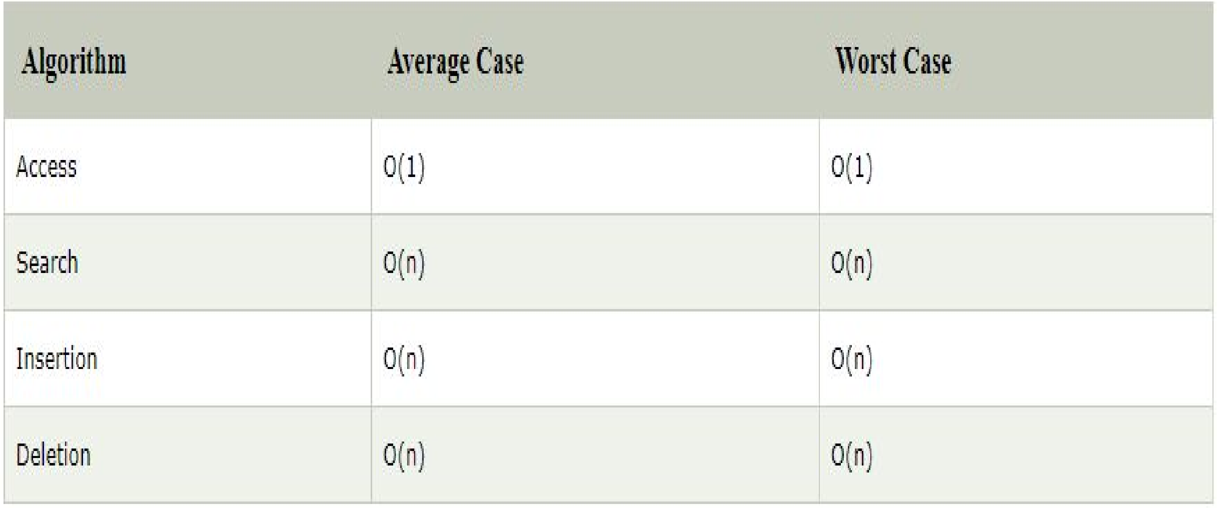
2- Copy Elements From Old Array To New Array Until We Reached To The Position Required.

3- Delete Element At this Position.

4- Copy And Shift Remaining Items.



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)

Space Complexity :

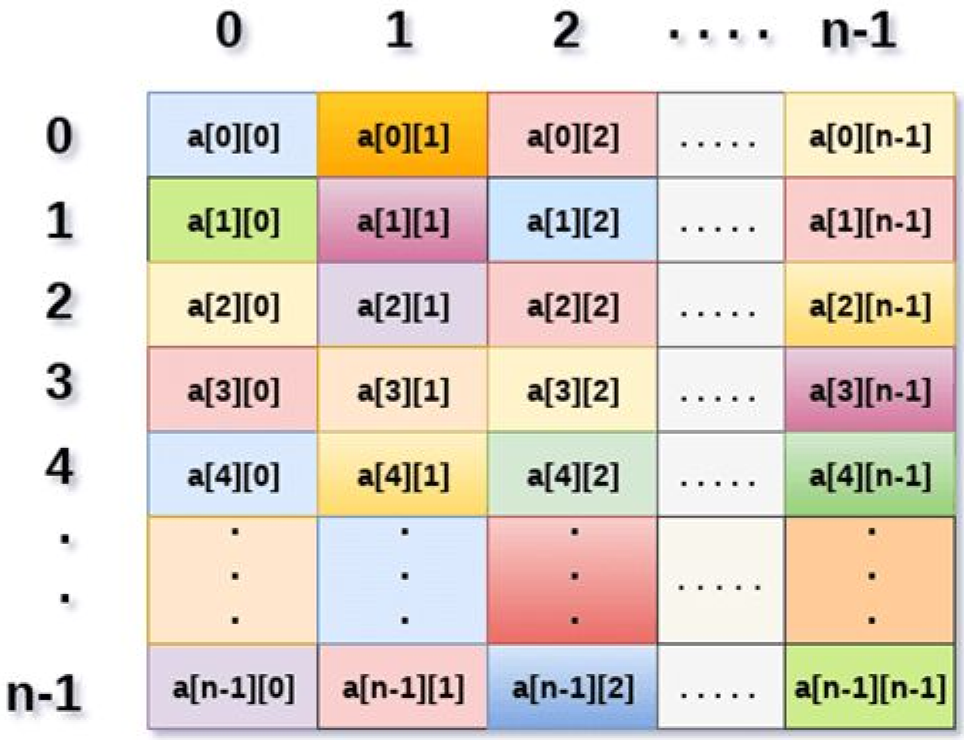
In array, space complexity for worst case is **O(n)**.

2D Array :

- What is 2D array ?
- How to declare 2D Array ?
- How do we access data in a 2D array ?
- How To Store 2D Array in the memory ?
- Calculating Address of element of a 2D array ?

What is 2D Array ?

- an array of arrays
- organized as matrices which can be represented as the collection of rows and columns.

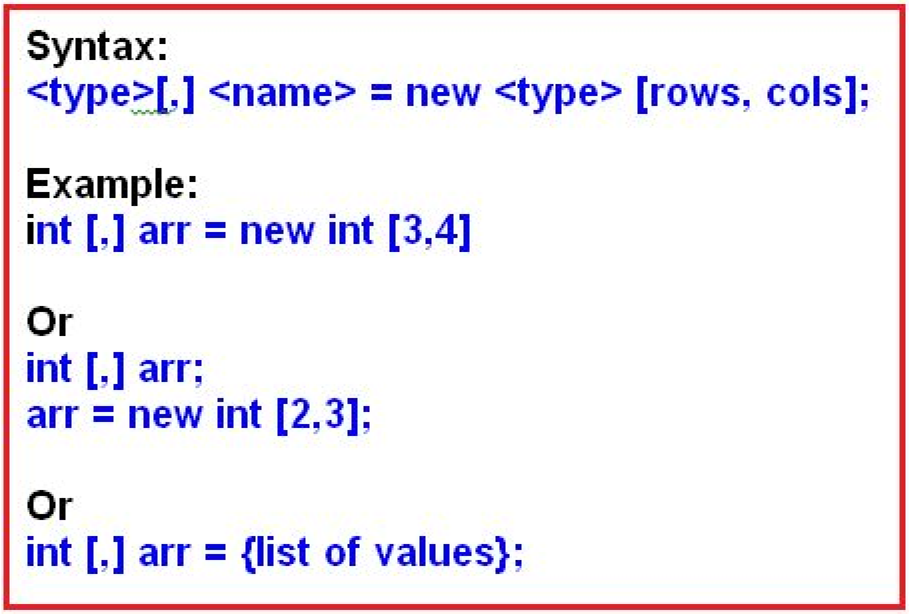


	0	1	2	. . . .	n-1
0	a[0][0]	a[0][1]	a[0][2]	. . . . .	a[0][n-1]
1	a[1][0]	a[1][1]	a[1][2]	. . . . .	a[1][n-1]
2	a[2][0]	a[2][1]	a[2][2]	. . . . .	a[2][n-1]
3	a[3][0]	a[3][1]	a[3][2]	. . . . .	a[3][n-1]
4	a[4][0]	a[4][1]	a[4][2]	. . . . .	a[4][n-1]
. .	. . .	. . .	. . .	. . . . .	. . .
n-1	a[n-1][0]	a[n-1][1]	a[n-1][2]	. . . . .	a[n-1][n-1]

a[n][n]

How to declare 2D Array ?

- In C# :



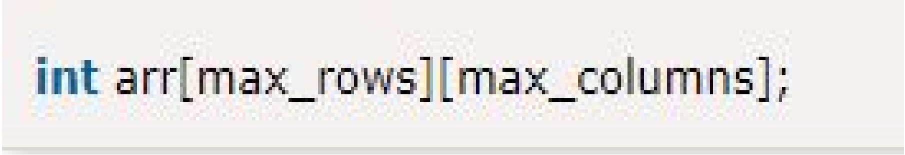
```
Syntax:
<type>[,] <name> = new <type> [rows, cols];

Example:
int [,] arr = new int [3,4]

Or
int [,] arr;
arr = new int [2,3];

Or
int [,] arr = {list of values};
```

- **In C++ :**

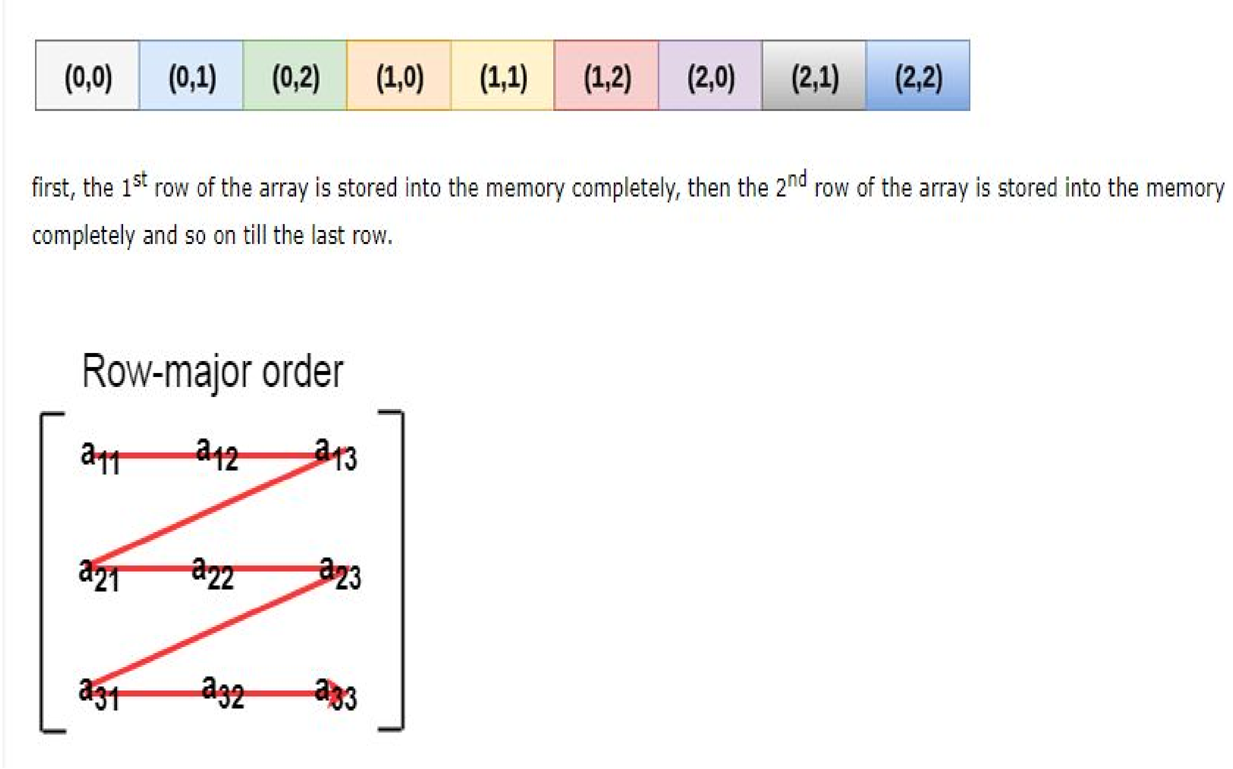


```
int arr[max_rows][max_columns];
```

How To Store 2D array into the memory

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

1. Row Major ordering (Row By Row)



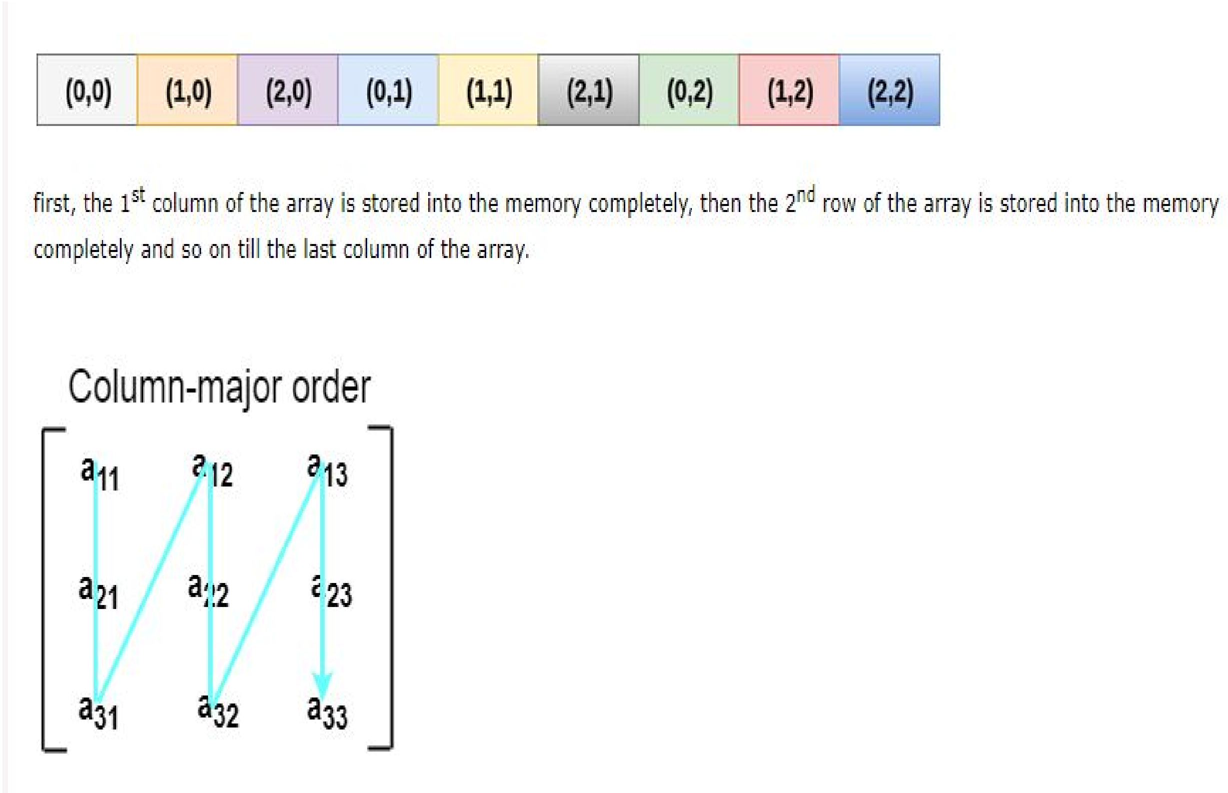
(0,0)	(0,1)	(0,2)	(1,0)	(1,1)	(1,2)	(2,0)	(2,1)	(2,2)

first, the 1st row of the array is stored into the memory completely, then the 2nd row of the array is stored into the memory completely and so on till the last row.

Row-major order

a_{11} a_{12} a_{13}
a_{21} a_{22} a_{23}
a_{31} a_{32} a_{33}

2. Column Major ordering (Column By Column)



(0,0)	(1,0)	(2,0)	(0,1)	(1,1)	(2,1)	(0,2)	(1,2)	(2,2)

first, the 1st column of the array is stored into the memory completely, then the 2nd row of the array is stored into the memory completely and so on till the last column of the array.

Column-major order

Resources :

1- https://www.javatpoint.com/data-structure-array

3- https://www.guru99.com/array-data-structure.html

4- https://beginnersbook.com/2018/10/data-structure-array/

5- http://www.mathcs.emory.edu/~cheung/Courses/170/Syllabus/09/basics.html

6- https://career.guru99.com/top-50-array-interview-questions-answers/

7- https://hackernoon.com/50-data-structure-and-algorithms-interview-questions-for-programmers-b4b1ac61f5b0