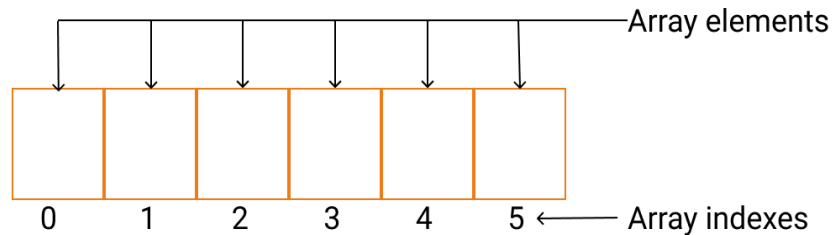


# Array

- An Array is a [Linear data structure](#)
- It is a collection of data items having similar data types stored in contiguous memory locations.
- By knowing the address of the first item we can easily access all items/elements of an array.
- Arrays and its representation is given below.



- Array Description:
  - **Array Index:** The location of an element in an array has an index, which identifies the element. Array index starts from 0.
  - **Array element:** Items stored in an array is called an element. The elements can be accessed via its index.
  - **Array Length:** The length of an array is defined based on the number of elements an array can store. In the above example, array length is 6 which means that it can store 6 elements.
- When an array of size and type is declared, the compiler allocates enough memory to hold all elements of data.
- E.g. an array `face [10]` will have 10 elements with index starting from 0 to 9 and the memory allocated contiguously will be 20 bytes (integer = 2 bytes).
- The compiler knows the address of the first byte of the array only. Also, the address of the first byte is considered as the memory address for the whole array.

## Types of Arrays

The various types of arrays are as follows.

- One dimensional array
- Multi-dimensional array
- **One-Dimensional Array**
  - A one-dimensional array is also called a single dimensional array
  - where the elements will be accessed in sequential order.
  - This type of array will be accessed by the subscript of either a column or row index.

- **Multi-Dimensional Array**

- When the number of dimensions specified is more than one, then it is called as a multi-dimensional array.
- Multidimensional arrays include 2D arrays and 3D arrays.

### Two-dimensional Array

		Columns			
		0	1	2	3
Rows	0	[0] [0]	[0] [1]	[0] [2]	[0] [3]
	1	[1] [0]	[1] [1]	[1] [2]	[1] [3]
	2	[2] [0]	[2] [1]	[2] [2]	[2] [3]

1st Subscript indicating the rows      2nd Subscript indicating the columns




- **E.g. A two-dimensional array**

- Accessed with subscript of row and column index
- For traversal the value of the rows and columns will be considered
- E.g. **face [3] [4]**, the first index specifies the number of rows and the second index specifies the number of columns and the array can hold 12 elements ( $3 * 4$ )

- **A three-dimensional array**

- The array **face [5] [10] [15]** can hold 750 elements ( $5 * 10 * 15$ ).

### Address Calculation in single (one) Dimension Array:

Actual Address of the 1 <sup>st</sup> element of the array is known as <b>Base Address (B)</b> Here it is 1100				Memory space acquired by every element in the Array is called <b>Width (W)</b> Here it is 4 bytes		
						
Actual Address in the Memory	1100	1104	1108	1112	1116	1120
Elements	15	7	11	44	93	20
Address with respect to the Array (Subscript)	0	1	2	3	4	5
						
Lower Limit/Bound of Subscript (LB)						

- Address of an element  $A[I]$  is calculated using the following formula:

- $1100 + (1-0) * 4$
- $1100 + (4-0) * 4$ 
  - **Address of A [ I ] = B + W \* ( I - LB )**
  - Where,
    - B** = Base address
    - W** = Storage Size of one element stored in the array (in byte)
    - I** = Subscript of element whose address is to be found
    - LB** = Lower limit / Lower Bound of subscript, if not specified assume 0 (zero)
  - **Example:**

**Base address of an array B[1300.....1900] as 1020 and size of each element is 2 bytes in the memory. Find the address of B[1700]**

**Solution:**

**Sol : B = 1020, LB = 1300, W = 2, I = 1700**

$$\begin{aligned}
 \text{Address of A [ I ]} &= \text{B} + \text{W} * (\text{I} - \text{LB}) \\
 &= 1020 + 2 * (1700 - 1300) \\
 &= 1020 + 2 * 400 \\
 &= 1020 + 800 \\
 &= 1820
 \end{aligned}$$

### Why does Array Indexing start with 0?

- Here **a** itself is a pointer which contains the memory location of the first element of the array
- first element can be accessed with **a[0]**
  - internally decoded by the compiler as **\*(a + 0)**.
- second element can be accessed by **a[1]** or **\*(a + 1)**.
- As **a** contains the address of the first element = Base Address
- and index describes the offset from the first element, i.e. the distance from the first element.
  - If array indexing starts at 1 instead of 0
  - the first element can be accessed by **a[1]**
    - which is internally decoded as **\*(a + 1 - 1)**.
- Thus we have to perform one extra operation i.e. subtraction by 1.
- This extra operation will greatly decrease the performance when the program is big.
- Thus to avoid this extra operation and improve the performance, array indexing starts at 0 and not at 1.

## Address Calculation in Double (Two) Dimensional Array:

- While storing the elements of a 2-D array in memory, these are allocated contiguous memory locations.
- Therefore, a 2-D array must be linearized so as to enable their storage.
- There are two alternatives to achieve linearization:
  - Row-Major
  - Column-Major.

		Column Index			
		0	1	2	3
Row Index	0	8	6	5	4
	1	2	1	9	7
	2	3	6	4	2

Two-Dimensional Array

### Row-Major (Row Wise Arrangement)

8	6	5	4	2	1	9	7	3	6	4	2
Row 0				Row 1				Row 2			
0	1	2	3	4	5	6	7	8	9	10	11

- Address of an element  $A[I][J] = B + W * [N * (I - L_r) + (J - L_c)]$  for the array declared as  $A[M][N]$

Where

**B** = Base address

**I** = Row subscript of element whose address is to be found

**J** = Column subscript of element whose address is to be found

**W** = Storage Size of one element stored in the array (in byte)

**L<sub>r</sub>** = Lower limit of row/start row index of matrix, if not given assume 0 (zero)

**L<sub>c</sub>** = Lower limit of column/start column index of matrix, if not given assume 0 (zero)

**M** = Number of row of the given matrix

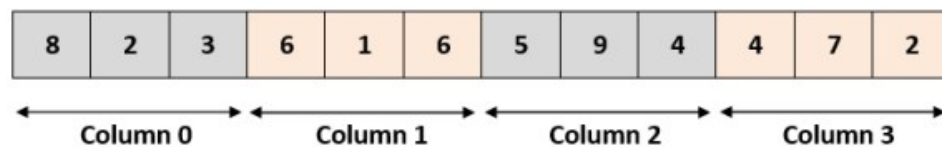
**N** = Number of column of the given matrix

Example : for the given A[4][4] with A[0..3][0..3]

- Address of A[1][1] =  $B + W * [4 * (1-0) + (1-0)]$   
=  $B + W * [5]$

## Column Oriented storage :

Column-Major (Column Wise Arrangement)



- Address of A[ I ][ J ] =  $B + W * [M * ( J - Lc ) + ( I - Lr )]$

Where

**B** = Base address

**I** = Row subscript of element whose address is to be found

**J** = Column subscript of element whose address is to be found

**W** = Storage Size of one element stored in the array (in byte)

**Lr** = Lower limit of row/start row index of matrix, if not given assume 0 (zero)

**Lc** = Lower limit of column/start column index of matrix, if not given assume 0 (zero)

**M** = Number of row of the given matrix

**N** = Number of column of the given matrix

- **Important Note :**

- Usually number of rows and columns of a matrix are given ( like A[20][30] or A[40][60] ) but if it is given as A[Lr- - - - Ur, Lc- - - - Uc]. In this case number of rows and columns are calculated using the following methods:
  - Number of rows (**M**) will be calculated as =  $(Ur - Lr) + 1$   
Number of columns (**N**) will be calculated as =  $(Uc - Lc) + 1$
  - And rest of the process will remain same as per requirement (Row Major Wise or Column Major Wise).

**Examples:**

**Q 1.** An array X [-15.....10, 15.....40] requires **one byte of storage**. If beginning location is 1500 determine the location of X [15][20].

**Q 1.** An array X [-15.....10, 15.....40] requires **one byte of storage**. If beginning location is 1500 determine the location of X [15][20].

**Solution:**

As you see here the number of rows and columns are not given in the question. So they are calculated as:

Number of rows say **M** = **(Ur – Lr) + 1** = [10 – (- 15)] +1 = 26

Number of columns say **N** = **(Uc – Lc) + 1** = [40 – 15)] +1 = 26

**(i) Row Major Wise Calculation of above equation**

The given values are: B = 1500, W = 1 byte, I = 15, J = 20, Lr = -15, Lc = 15, N = 26

$$\begin{aligned}\text{Address of A [ I ][ J ]} &= B + W * [ N * ( I - Lr ) + ( J - Lc ) ] \\ &= 1500 + 1 * [ 26 * ( 15 - (-15) ) + ( 20 - 15 ) ] \\ &= 1500 + 1 * [ 26 * 30 + 5 ] \\ &= 1500 + 1 * [ 780 + 5 ] \\ &= 1500 + 785 \\ &= 2285 \text{ [Ans]}\end{aligned}$$

**Q 1.** An array X [-15.....10, 15.....40] requires **one byte of storage**. If beginning location is 1500 determine the location of X [15][20].

**Solution:**

As you see here the number of rows and columns are not given in the question. So they are calculated as:

Number of rows say **M** = **(Ur – Lr) + 1** = **[10 – (- 15)] + 1 = 26**

Number of columns say **N** = **(Uc – Lc) + 1** = **[40 – 15)] + 1 = 26**

**(i) Row Major Wise Calculation of above equation**

The given values are: B = 1500, W = 1 byte, I = 15, J = 20, Lr = -15, Lc = 15, N = 26

$$\text{Address of A [ I ][ J ]} = B + W * [ N * ( I - Lr ) + ( J - Lc ) ]$$

$$\begin{aligned} &= 1500 + 1 * [ 26 * ( 15 - (-15) ) + ( 20 - 15 ) ] \\ &= 1500 + 1 * [ 26 * 30 + 5 ] \\ &= 1500 + 1 * [ 780 + 5 ] \\ &= 1500 + 785 \\ &= 2285 \text{ [Ans]} \end{aligned}$$

**(ii) Column Major Wise Calculation of above equation**

The given values are: B = 1500, W = 1 byte, I = 15, J = 20, Lr = -15, Lc = 15, M = 26

$$\text{Address of A [ I ][ J ]} = B + W * [ ( I - Lr ) + M * ( J - Lc ) ]$$

$$\begin{aligned} &= 1500 + 1 * [ ( 15 - (-15) ) + 26 * ( 20 - 15 ) ] \\ &= 1500 + 1 * [ 30 + 26 * 5 ] \\ &= 1500 + 1 * [ 160 ] \\ &= 1660 \text{ [Ans]} \end{aligned}$$



**Total memory allocated to an Array = Number of elements \* size of one element**

### **Single Dimension:**

Total memory allocated for an Integer Array of N elements

= Number of elements \* size of one element

=  $N * 4$  bytes

=  $10 * 4$  bytes = **40 Bytes**, where  $N = 10$

=  $500 * 4$  bytes = **2000 Bytes**, where  $N = 500$

Total memory allocated for an character Array of N elements

= Number of elements \* size of one element

=  $N * 1$  Byte

=  $10 * 1$  Byte = **10 Bytes**, where  $N = 10$

=  $500 * 1$  Byte = **500 Bytes**, where  $N=500$

### **Two Dimensions :**

**Total memory allocated for 2D Array**

= Number of elements \* size of one element

= Number of Rows \* Number of Columns \* Size of one element

**Total memory allocated for an Integer Array of size M X N**

= Number of elements \* size of one element

=  $M \text{ Rows} * N \text{ Columns} * 4$  Bytes

=  $10*10 * 4$  bytes = **400 Bytes**, where  $M = N = 10$

=  $500*5 * 4$  bytes = **10000 Bytes**, where  $M=500$  and  $N= 5$

**Total memory allocated for a character Array of N elements**

= Number of elements \* size of one element

=  $M \text{ Rows} * N \text{ Columns} * 1$  Byte

=  $10*10 * 1$  Byte = **100 Bytes**, where  $N = 10$

=  $500*5 * 1$  Byte = **2500 Bytes**, where  $M=500$  and  $N= 5$

**Example 2:**

Each element of an array  $\text{arr}[15][20]$  requires 'W' bytes of storage. If the address of  $\text{arr}[6][8]$  is 4440 and the base address at  $\text{arr}[1][1]$  is 4000, find the width 'W' of each cell in the array  $\text{arr}[][]$  when the array is stored as Column Major Wise.

**Solution: Example 2:**

Each element of an array **arr[15][20]** requires 'W' bytes of storage. If the **address of arr[6][8] is 4440** and the **base address at arr[1][1]** is 4000, find the width 'W' of each cell in the array arr[][] when the array is stored as **Column Major Wise**.

Given :

**B** = Base address = 4000

**I** = Row subscript of element whose address is to be found = 6

**J** = Column subscript of element whose address is to be found = 8

**W** = Storage Size of one element stored in the array (in byte) = NOT Given

**Lr** = Lower limit of row/start row index of matrix = 1

**Lc** = Lower limit of column/start column index of matrix = 1

**M(or R)** = Number of row of the given matrix = 15

**N (or C)** = Number of column of the given matrix = 20

### Solution: Example 2:

Each element of an array **arr[15][20]** requires 'W' bytes of storage. If the **address of arr[6][8]** is **4440** and the **base address at arr[1][1]** is 4000, find the width 'W' of each cell in the array arr[][] when the array is stored as **Column Major Wise**.

Given :

**B** = Base address = 4000

**I** = Row subscript of element whose address is to be found = 6

**J** = Column subscript of element whose address is to be found = 8

**W** = Storage Size of one element stored in the array (in byte) = NOT Given

**L<sub>r</sub>** = Lower limit of row/start row index of matrix = 1

**L<sub>c</sub>** = Lower limit of column/start column index of matrix = 1

**M(or R)** = Number of row of the given matrix = 15

**N (or C)** = Number of column of the given matrix = 20

Address of [I, J]<sup>th</sup> element in column-major

$$= B + W[R(J - L_c) + (I - L_r)]$$

$$\Rightarrow 4440 = 4000 + W[15(8 - 1) + (6 - 1)]$$

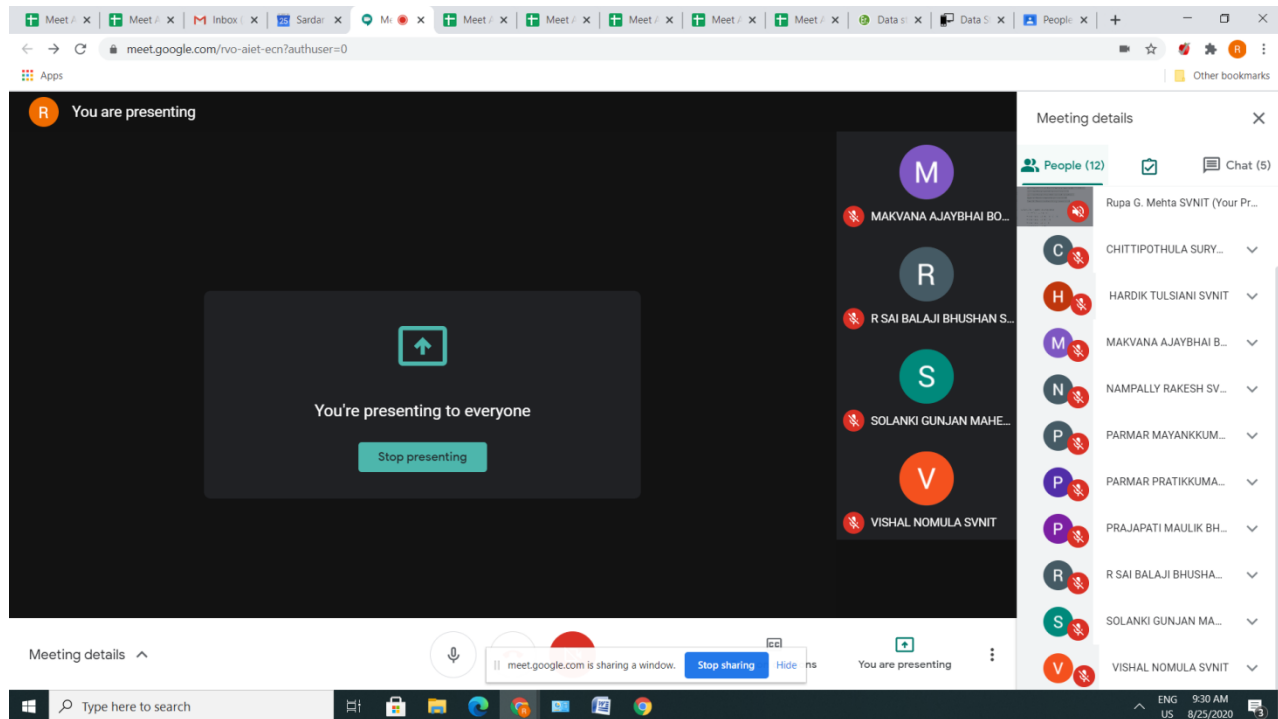
$$\Rightarrow 4440 = 4000 + W[15(7) + 5]$$

$$\Rightarrow 4440 = 4000 + W[105 + 5]$$

$$\Rightarrow 4440 = 4000 + W[110]$$

$$\Rightarrow W[110] = 440$$

$$\Rightarrow W = 4.$$



### Example 3:

A matrix  $ARR[-4 \dots 6, 3 \dots 8]$  is stored in the memory with each element requiring 4 bytes of storage. If the base address is 1430, find the address of  $ARR[3][6]$  when the matrix is stored in Row Major Wise.

Given :

**B** = Base address = 1430

**I** = Row subscript of element whose address is to be found = 3

**J** = Column subscript of element whose address is to be found = 6

**W** = Storage Size of one element stored in the array (in byte) = 4

**Lr** = Lower limit of row/start row index of matrix = -4

**Lc** = Lower limit of column/start column index of matrix = 3

**M(or R)** = Number of row of the given matrix =  $6 - (-4) + 1 = 11$

**N (or C)** = Number of column of the given matrix =  $8 - 3 + 1 = 6$

**Example 3:**

A matrix  $ARR[-4 \dots 6, 3 \dots 8]$  is stored in the memory with each element requiring 4 bytes of storage. If the base address is 1430, find the address of  $ARR[3][6]$  when the matrix is stored in Row Major Wise.

Given :

**B** = Base address = 1430

**I** = Row subscript of element whose address is to be found = 3

**J** = Column subscript of element whose address is to be found = 6

**W** = Storage Size of one element stored in the array (in byte) = 4

**L<sub>r</sub>** = Lower limit of row/start row index of matrix = -4

**L<sub>c</sub>** = Lower limit of column/start column index of matrix = 3

**M(or R)** = Number of row of the given matrix =  $6 - (-4) + 1 = 11$

**N (or C)** = Number of column of the given matrix =  $8 - 3 + 1 = 6$

**Address of [I, J]<sup>th</sup> element in row-major**

$$= B + W[C(I - L_r) + (J - L_c)]$$

$$\Rightarrow \text{Address of } ARR[3][6] = 1430 + 4[6(3 - (-4)) + (6 - 3)]$$

$$\Rightarrow \text{Address of } ARR[3][6] = 1430 + 4[6(3 + 4) + 3]$$

$$\Rightarrow \text{Address of } ARR[3][6] = 1430 + 4[6(7) + 3]$$

$$\Rightarrow \text{Address of } ARR[3][6] = 1430 + 4[42 + 3]$$

$$\Rightarrow \text{Address of } ARR[3][6] = 1430 + 4[45]$$

$$\Rightarrow \text{Address of } ARR[3][6] = 1430 + 180$$

$$\Rightarrow \text{Address of } ARR[3][6] = 1610.$$