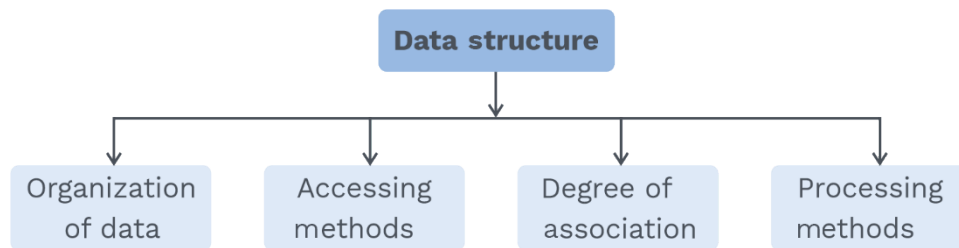
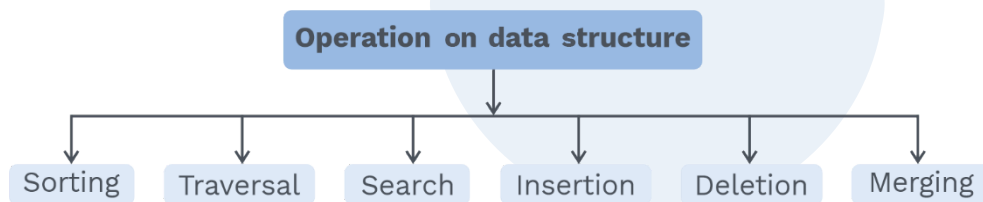
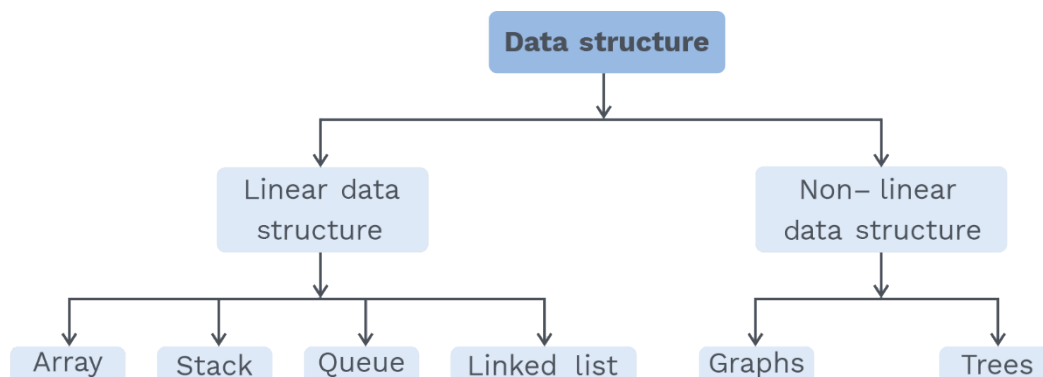


DATA STRUCTURES

Way of organizing data in computer memory so that the memory can be efficiently used in terms of both time and space.

**Operations on data structures:**

- 1) Traversal: Visiting each element in the list
- 2) Search: Finding the location of an element with a given value of the record with a given key.
- 3) Insertion: Adding a new element to the list.
- 4) Deletion: Removing an element from the list.
- 5) Sorting: Arranging the elements.
- 6) Merging: Combining two lists into one.

Types of data structures:

**Linear data structure:**

A Linear relationship between the elements is represented by means of contiguous memory locations. Another way of representing this linear relationship is by using pointers or linked list.

E.g. Array, stack, queue, linked list

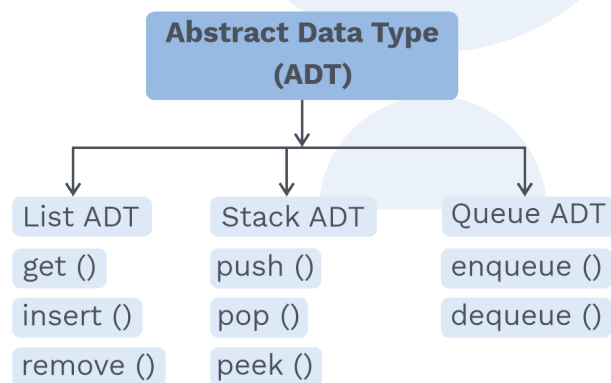
Non-linear data structures:

Non-linear data structures are not arranged sequentially in the memory. The elements in non-linear data structures cannot be traversed in a single run. Although they are more memory efficient than linear data structures.

E.g. graphs, trees.

Abstract data type (ADT):

- It is a class or datatype whose objects are defined by a set of values and a set of operations
- Combining data structure with its operation

**Previous Years' Question**

An Abstract Data Type (ADT) is :

- a) same as an abstract class
- b) a data type that cannot be instantiated
- c) a data type for which only the operations defined on it can be used, but nothing else.
- d) all of the above

Sol: c)

(GATE 2005 : 1-Mark)



2.1 BASICS OF ARRAY

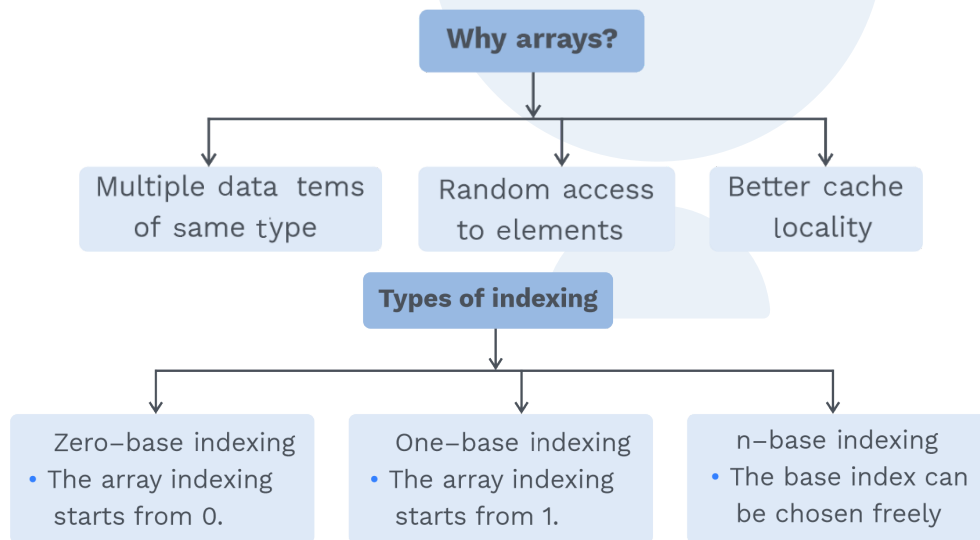
Introduction:

- An array is a collection of similar types of data items.
- Each data item is called an element of an array
- Often referred to as homogeneous collection of data elements.
- The datatype of elements may be any valid data type like char, int or float.
- The elements of the an array share same variable name, but each variable has a different index number. This index number is called the subscript.

Syntax: datatype array_name [Size];

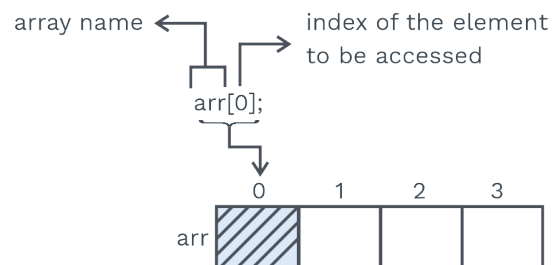
E.g. int arr[10]; //represents an array of 10 elements each of integer datatype.

- The array index starts from 0 in C language; hence arr[0] represents the first element of the array.



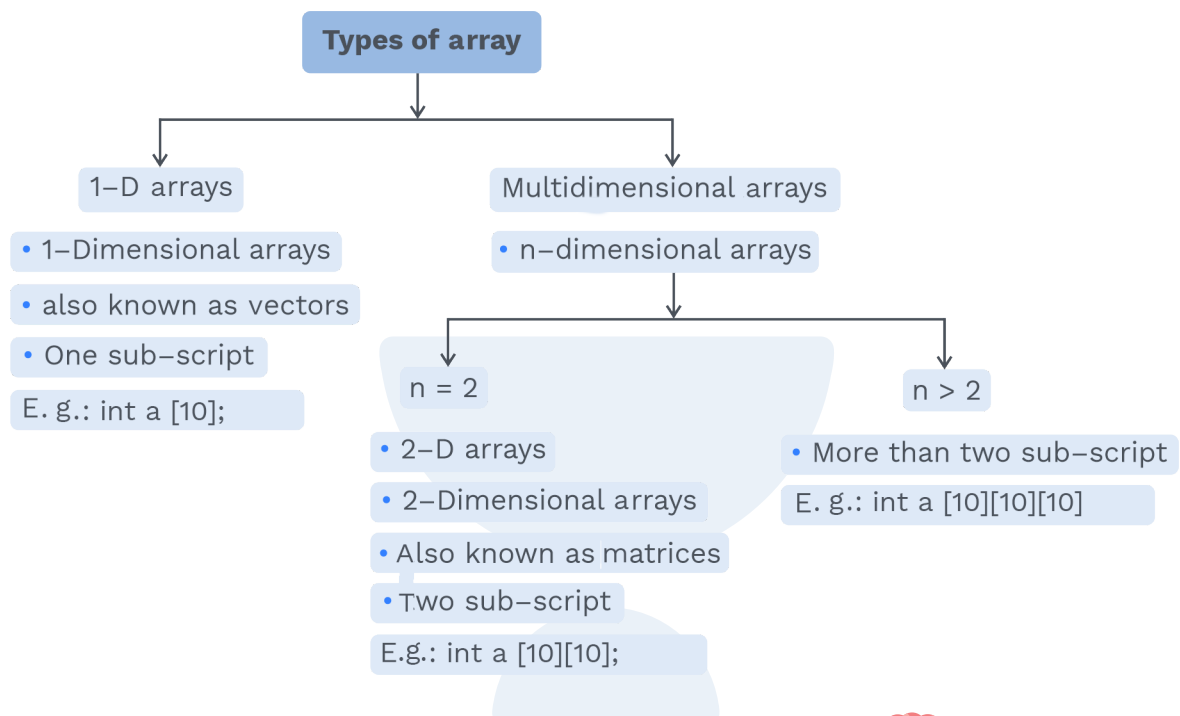
Note:

Programming languages that allow n-base indexing, also allow negative base index values.





- One disadvantage of arrays is that they have fixed sizes. Once an array is declared, its size cannot be changed. This is due to static memory allocation.

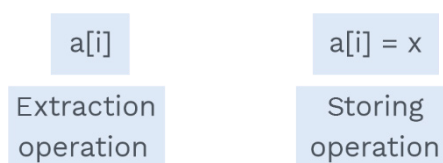


Rack Your Brain

What is the purpose of using multi-dimensional arrays?

Properties of array:

- 1) A one-dimensional array is a linear data structure which is used to store similar types of elements.
- 2) The two basic operations that access an array are:
 - i) Extraction operation: It is a function that accepts an array 'a' and an index 'i' and returns an element of the array.
 - ii) Storing operation: It accepts an array 'a', an index 'i' and an element 'x'.





- 3) The smallest element of an array's index is called the lower bound. In C lower bound is always 0.
- 4) The highest element is called the upper bound and is usually one less than the number of elements.
- 5) The number of elements in the array is called range.

$$\text{Range} = \text{Upper bound} - \text{lower bound} + 1$$

E.g. An array 'a' whose lower bound is '0' and the upper bound is 99, then the range would be

$$\Rightarrow 99 - 0 + 1$$

$$\Rightarrow 100$$

- 6) Array elements are stored at subsequent memory locations
- 7) 2-D arrays are by default stored in row- major order.
- 8) In C Programming, the lowest index of a 1-D array is always zero, while the upper index is the size of the array minus one.
- 9) Array name represents the base address of the array.
- 10) Only constants and literal values can be assigned as a value of a number of elements.

E.g.

```
const int MAX = 100 ;  
int a[MAX] ;  
int b[100] ;
```

VALID DECLARATION

```
int MAX = 100 ;  
int a[MAX] ;
```

INVALID DECLARATION

Different types of array:

1-D arrays

- One-dimensional array is used when it is necessary to keep a large number of items in memory and reference all the items in a uniform manner.

Syntax: `int b[100];`

- This declaration reserves 100 successive memory locations, each containing a single integer value.
- The address of the first location is called the base address and denoted by `base(b)`.
- The size of each element be `w` then address of an element `i` in an array `b` is given as:

$$b[i] = \text{base}(b) + i * w$$



Address	
100	126
101	127
102	128
103	129
.	.
.	.
.	.
.	.
n	

An array data of element



Rack Your Brain

Which of the following is an absolute array declaration in C?

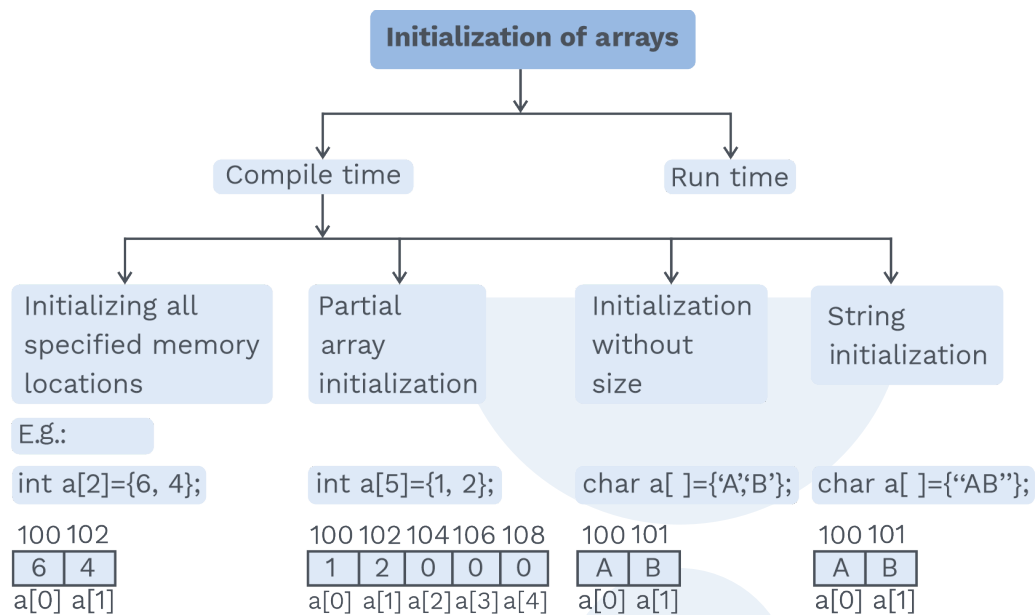
- I) `int a[3];`
- II) `int a[3] = {1, 2, 3}`
- III) `int a[3] = { };`
- IV) `int a[3] = {0};`
- V) `int a[3] = {[0..1] = 3};`
- VI) `int a[] = {[0..1] = 3};`
- VII) `int *a ;`

E.g. Program to print the average of 100 integers and how much each deviates from the average.

```
#include <stdio.h>
#define Num 100
average( )
{
    int n[Num];
    int i ; int total = 0;
    float avg, diff;
    for (i = 0; i < Num ; i++)
    {
        scanf(“ %d”, & n[i]);
        total += n[i];
    }
    avg = total/Num ;
    printf(“Number difference”);
    for (i = 0; i < Num ; i++)
    {
        diff = n[i] - avg;
        printf ( “\n %d %f”, n[i], diff);
    }
}
```



```
printf ("\n average is: %f", avg));
main()
{   average();
}
```



2-D arrays

- A 2-dimensional array is a logical data structure helpful in solving problems.

Syntax: `int a[3][5];`

here [3] is the number of rows, and [5] is the number of columns.

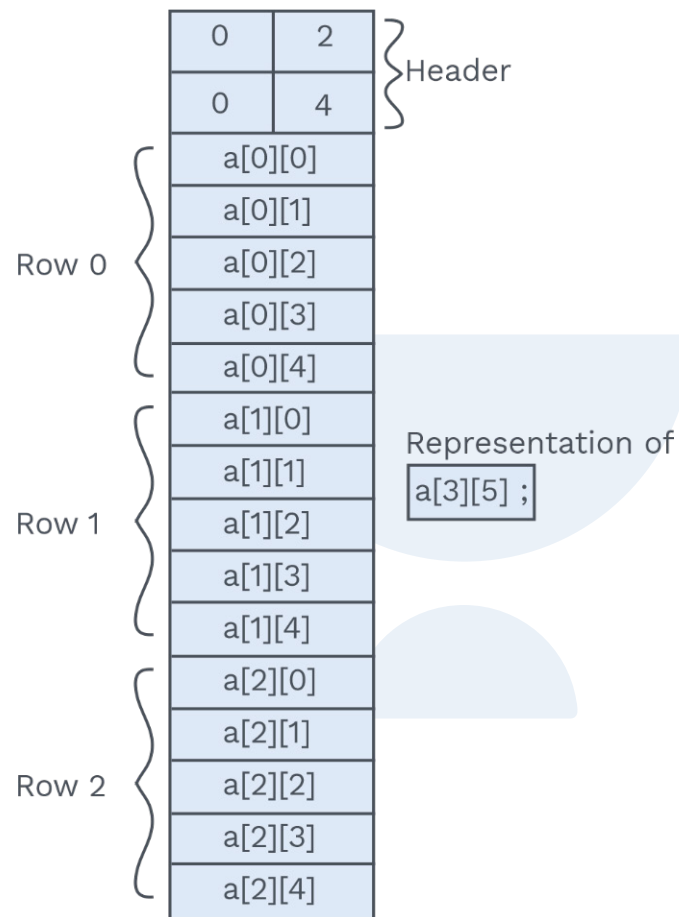
- This represents a new array containing 3 elements. Each of these elements in itself is an array containing five integers.
- The 2-D array has 2-indices called row and column.
- The number of rows or columns is called the range of the dimension.

	Column	Column	Column	Column	Column
	0	1	2	3	4
Row 0					
Row 1					
Row 2					

a[1][3] ;



- The arrays are, by default stored in row-major order. i.e. the first row of the array occupies the first set of memory locations, the second row occupies the next set so on and so forth.



Accessing element $a[i][j]$ from a 2-D array $a[m][n]$ then,
(Here, starting index of array is 0)

Row-major order:

$$a[i][j] = \text{base}(a) + (i * n + j) * w$$

Column-major order:

$$a[i][j] = \text{base}(a) + (i + j * m) * w$$

Where:

m : no. of rows

n : no. of columns

w : element size

base (a): base address of array $a[m][n]$



Initialization of 2D arrays

- One can initialize a 2-D array in the form of a matrix.

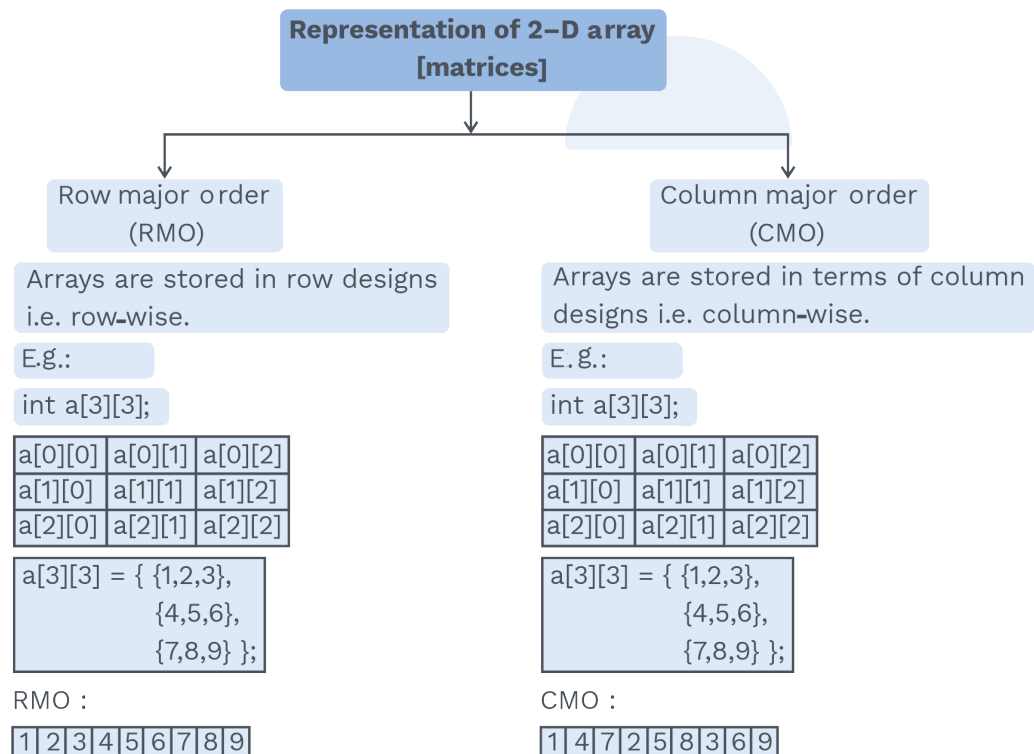
E.g. `int a[2][3] = { {0, 0, 0},
 {1, 1, 1} };`

- When the array is completely initialized with all the values, then we do not need to specify the size of the first dimension.

E.g. `int a[][3] = { {0, 0, 3},
 {1, 1, 1} };`

Note:

- If the values are missing in an initialization, then they are by default considered as 0.
- The first dimension is optional but the second dimension must be specified. i.e. rows are optional in a 2-D array, but initialization of columns is mandatory in 2-D array initialization.

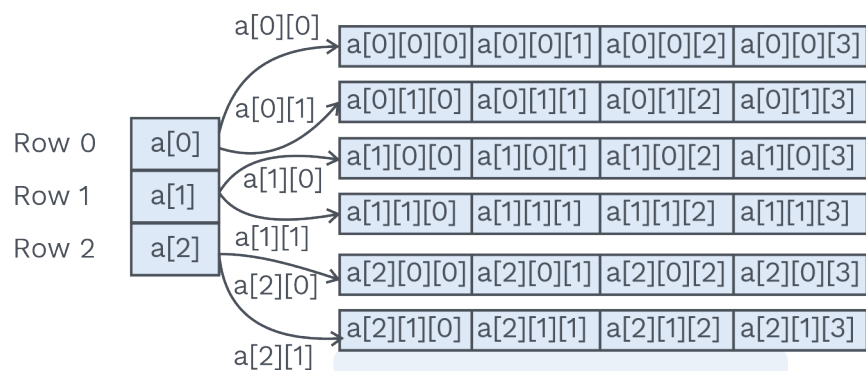




Multi-dimensional arrays

- Arrays with more than two dimensions

Syntax: `int a[3][2][4] ;// 3-D array`



- The first subscript specifies the plane number, the second specifies the row number, and the third specifies the column number.

`int a[p][m][n] ;`

p: Plane number

m: row number

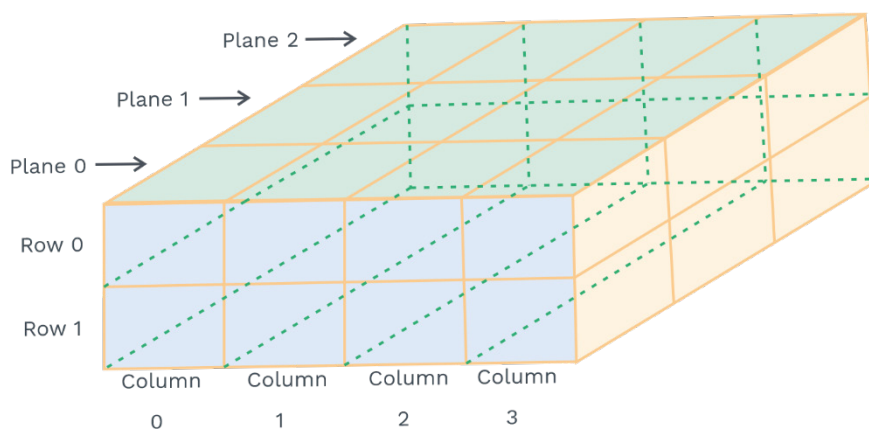
n: column number

E.g. `a[3][2][4] ;`

no. of plane = 3

no. of rows = 2

no. of columns = 4



Note:

C does allow an arbitrary number of dimensions.

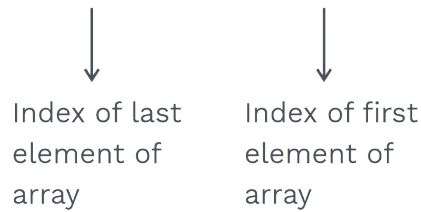
E.g.: There can be a 6-dimensional array which has six subscripts.



2.2 ACCESS ELEMENT

1-D array:

Number of elements = Upper bound – Lower bound + 1



- Let the base address of an array a be b and the lower bound be represented as $L.B$, and this size of each element be w .
- Then the address of K^{th} element of an array a is given as:

$$a[K] = b + [K - L.B] * w$$

- If array indexing starts from 0 then:

$$a[K] = b + K * w$$

SOLVED EXAMPLES

Q1 Let the base address of the first element of the array be 200 and each element occupy 4 bytes in the memory, then the address of the fifth element of a 1-D array $a[10]$?

Sol: Given,
base address (b) = 200
size of each element (w) = 4 bytes

	0	1	2	3	4	5	6	7	8	9
$a[10]$										

since $L.B$ not given, it is by default 0. Here the fifth element refers to the element at array index 4

fifth element = $a[4]$

$\therefore K = 4$

$a[K] = b + K * w$

$a[4] = 200 + 4 * 4$

$a[4] = 216$



Q2 Consider an array declaration as $A[-7...7]$ where each element is of 4 bytes; if the base address of the array is 3000, then find the address of $A[0]$?

Sol: Given,
array : $A[-7...7]$
 $W = 4$ bytes
 $b = 3000$
no. of elements $= U.B - L.B + 1$
 $= 7 - (-7) + 1$
 $= 14 + 1$
 $= 15$
 $\therefore A[K] = b + [(K - L.B)] * w$
 $A[0] = 3000 + [(0 - (-7))] * 4$
 $= 3000 + 28$
 $A[0] = 3028$
OR
Since: $A[-7...7] = A[0...14]$
 $\therefore A[0] = A[0 - (-7)]$
 $A[0] = A[7]$
 $\therefore A[7] = b + K * w$
 $= 3000 + 7 * 4$
 $A[7] = 3028$

Previous Years' Question (GATE 2005: 2-Mark)



A program P reads in 500 integers in the range $[0...100]$ representing the scores of 500 students. It then prints the frequency of each score above 50. What would be the best way for P to store the frequencies?

- a) An array of 50 numbers
- b) An array of 100 numbers
- c) An array of 500 numbers
- d) A dynamically allocated array of 550 numbers

Sol: a)

Rack Your Brain



- 1) Let the base address of the first element of an array be 100, and each element occupy 2 bytes in the memory, then what is the address of $a[5]$.
- 2) Let an array be $a[-12...10]$ where each element occupies 3 bytes, then if the base address of the array a is 2000 then, what is the address of element $a[7]$?

**2-D array:**

Two ways of array implementation

Row major order

Let an array $A[m][n]$, then address of element $A[i][j]$ in row major order is given by:

$$A[i][j] = b + (i * n + j) * w$$

when: $A[0.....m-1][0.....n-1]$

$$A[i][j] = b + [(i-1) * n + (j-1)] * w$$

when: $A[1.....m][1.....n]$

Similarly:

$$A[L_1.....U_1][L_2.....U_2]$$

then:

$$A[i][j] = b + [(U_2 - L_2 + 1)(i - L_1) + (j - L_2)] * w$$

Where:

L_1 : lower bound of rows

U_1 : upper bound of rows

L_2 : lower bound of columns

U_2 : upper bound of columns

SOLVED EXAMPLES

Q3 Let an array $A[5.....15, -8.....8]$ be $A[11][17]$ stored in row major order. Then if the base address of the array is 500 and each element occupies 3 bytes of memory, what would be the address of $A[8][5]$?

Sol: given,
 $A[11][17]$ which is $A[5.....15, -8.....8]$
 $m = 11$ //no. of rows
 $n = 17$ //no. of columns
 $b = 500, w = 3$ bytes
or
 $L_1 = 5, L_2 = -8$
 $U_1 = 15, U_2 = 8$



this means $\Rightarrow A[0.....10] [0.....16]$

\downarrow \downarrow
 11 rows 17 columns

$$\text{i.e. } A[8][5] = A[8-(-5)] [5-(-8)] \\ \approx A[3][13]$$

$$A[i][j] = b + (i * n + j) * w$$

$$A[3][13] = 500 + (3 * 17 + 13) * 3 \\ = 500 + 192$$

$$A[3][13] = 692$$

or

$$A[8][5] = b + [(U_2 - L_2 + 1) (i - L_1) + (j - L_2)] * w \\ = 500 + [(8 - (-8) + 1) (8 - 5) + (5 - (-8))] * 3 \\ = 500 + [17 * 3 + 13] * 3 \\ = 500 + 192$$

$$A[8][5] = 692$$

Q4 Consider a 2-D array Arr of range $[-5.....5, 3.....13]$ where each element occupies 4 four memory cells, and the base address of the array is 200. Then what would be the address of location arr[5][6]?

Sol: Given,

$$\text{Arr}[-5.....5, 3.....13], b = 2000$$

$$\text{Arr}[0.....10, 0.....10], w = 4\text{bytes}$$

\downarrow \downarrow
 11 rows 11 columns

1st Approach:

$$\text{Arr}[-5.....5, 3.....13] \\ L_1 \quad U_1 \quad L_2 \quad U_2$$

$$A[i][j] = b + [(U_2 - L_2 + 1) (i - L_1) + (j - L_2)] * w$$

$$A[5][6] = 2000 + [(13 - 3 + 1) (5 - (-5)) + (6 - 3)] * 4 \\ = 2000 + [(10 * 11) + 3] * 4 \\ = 2000 + 452$$

$$A[5][6] = 2452$$

2nd Approach:

$$\text{Arr}[0.....10, 0.....10]$$



$$\begin{aligned}A[5][6] &\approx [5-1(-5)] [6-3] \\&\approx A[10] [3] \\A[10][3] &= b + (i*n + j) * w \\&= 2000 + ((10 \times 11) + 3) * 4 \\&= 2000 + 452 \\A[10][3] &= 2452\end{aligned}$$

Previous Years' Question (GATE 2000: 1-Mark)

An $n \times n$ array v is defined as follows :

$v[i, j] = i - j$ for all $i, j, 1 \leq i \leq n, 1 \leq j \leq n$

The sum of elements of the array v is

- a) 0 b) $n-1$ c) n^2-3n+2 d) $n^2(n+1)/2$

Sol: a)

Previous Years' Question (GATE 2014 : 1-Mark)

Let A be a square matrix of size $n \times n$. Consider the following program. What is the expected output?

$C=100$;

for $i = 1$ to n do

for $j = 1$ to n do

{

$Temp = A[i][j] + C$;

$A[i][j] = A[j][i]$;

$A[j][i] = Temp - C$;

}

for $i = 1$ to n do

for $j = 1$ to n do

 output ($A[i][j]$) ;

a) The matrix A itself

d) Transpose of matrix A

c) Adding 100 to the upper diagonal elements and subtracting 100 from lower diagonal elements of A

d) None of the above

Sol: a)

**Previous Years' Question (GATE 2015 Set-2 : 2-Marks)**

A Young tableau is a 2-D array of integers increasing from left to right and from top to bottom. Any unfilled entries are marked with ∞ , and hence there cannot be any entry to the right of, or below a ∞ . The following Young tableau consists of unique entries.

1	2	5	14
3	4	6	23
10	12	18	25
31	∞	∞	∞

When an element is removed from a Young tableau, other elements should be moved into its place so that the resulting table is still a Young tableau (unfilled entries may be filled with a ∞). The minimum number of entries (other than 1) to be shifted, to remove one from the given young tableau is

Sol: 5

Previous Years' Question (GATE 1998 : 2-Marks)

Let A be a two-dimensional array declared as follows:

A : array [1...10] [1...15] of integer;

Assuming that each integer takes one memory location. The array is sorted in row-major order and first element of the array is stored at location 100. What is the address of the element A[i][j]?

- a) $15i + j + 84$ b) $15j + i + 84$ c) $10i + j + 89$ d) $10j + i + 89$

Sol: a)

Column-major order

Let any array A[m][n] then address of element A[i][j] in column major order is given by:

$$A[i][j] = b + (i + j * m) * w$$

when: A[0.....m-1] [0.....n-1]

$$A[i][j] = b + ((i-1) + (j-1) * m) * w$$

when: A[1.....m] [1.....n]



Similarly: $A[L_1, \dots, U_1] [L_2, \dots, U_2]$
then:

$$A[i][j] = b + [(i - L_1) + (j - L_2)(U_1 - L_1 + 1)] * w$$

where:

L_1 : Lower bound of rows

U_1 : Upper bound of rows

L_2 : Lower bound of columns

U_2 : Upper bound of columns



Rack Your Brain

- 1) is the logical and mathematical model of a particular organization of data.
(a) Structure (b) Variables (c) Function (d) Data structures
- 2) Which of the following is not a primitive data structure?
a) Boolean b) Integer c) Arrays d) Character

SOLVED EXAMPLES

Q5 An array $A[20][30]$ is stored column-wise with, each element occupying 4 bytes of memory space. If the base address is 100, then what is the memory location of the element $A[5][15]$?

Sol: Given,
 $A[20][30]$ stored in column major order
 $w = 4$ bytes
 $b = 100$
 $A[5][15] = ?$
 $A[i][j] = b + (i + j * m) * w$
 $A[5][15] = 100 + (5 + 15 \times 20) * 4$
 $A[5][15] = 1320$



Q6 An array $A[-2....8] [-2....5]$ is stored in the memory in column major order such that each element occupies 6 bytes by memory. What would be the memory locations of $A[3][2]$? Consider the base address to be 5000.

Sol: Given,
 $A[-2....8] [-2....5]$
 $b = 5000$
 $A[i][j] = b + [(i - L_1) + (j - L_2) (U_1 - L_1 + 1)] * w$
 $A[3][2] = 5000 + [(3 - (-2)) + (2 - (-2)) (8 - (-2) + 1)] * 6$
 $= 5000 + [5 + 44] * 6$
 $= 5000 + 294$
 $A[3][2] = 5294$
 or
 Convert $A[-2....8] [-2....5]$ to 0 indexing.
 i.e. $A[0....10] [0....7]$ then $A[3][2]$ also
 maps to $A[5][4]$, $m = 11$, $n = 8$
 $\therefore A[5][4] = b + (i + j * m) * w$
 $= 5000 + (5 + 4 * 11) * 6$
 $= 5000 + 294$
 $A[5][4] = 5294$

Multi-dimensional arrays:

- In multi-dimensional structure the terms row and columns cannot be used, since there are more than 2 dimensions.

\therefore Let a N-Dimensional array:

$$A[L_1....U_1][L_2....U_2][L_3....U_3][L_4....U_4].....[L_N....U_N]$$

then location of $A[i, j, k, \dots, x]$ is given by:

$$= b + \{(i - L_1) [(U_2 - L_2 + 1) (U_3 - L_3 + 1) (U_4 - L_4 + 1) \dots (U_N - L_N + 1)] \\ + (j - L_2) [(U_3 - L_3 + 1) (U_4 - L_4 + 1) \dots (U_N - L_N + 1)] \\ + (k - L_3) [(U_4 - L_4 + 1) \dots (U_N - L_N + 1)] \dots + (x - L_N)\} * w$$

3-D Array

$$\text{Let a 3-D array } A[L_1....U_1][L_2....U_2][L_3....U_3]$$

then location of $A[i][j][k]$ is given by:

$$A[i][j][k] = b + \{(i - L_1) [(U_2 - L_2 + 1) (U_3 - L_3 + 1)] \\ + (j - L_2) [(U_3 - L_3 + 1)] \\ + (k - L_3)\} * w$$



SOLVED EXAMPLES

Q7 Suppose a 3-D array $A[2...8, -4...1, 6...10]$ has elements each occupying 4 bytes of memory, if the base address of the array is 100, then what would be the address of element at $A[6][2][5]$?

Sol: Given,

$$A[2...8, -4...1, 6...10], b = 100, w = 4 \text{ bytes}$$

↓
 L_1

↓
 U_1

↓
 L_2

↓
 U_2

↓
 L_3

↓
 U_3

then:

$$\begin{aligned}
 A[i][j][k] &= b + \{(i-L_1) [(U_2-L_2+1) (U_1-L_1+1)] \\
 &+ (j-L_2) [(U_1-L_1+1)] \\
 &+ (k-L_3)\} * w \\
 A[6][2][5] &= 100 + \{(6-(2)) [(1-(-4)+1) (8-2+1)] \\
 &+ (2-(-4)) [(8-2+1)] \\
 &+ (5-6)\} * 4 \\
 &= 100 + \{(4*6*7) + (6*7) + (-1)\} * 4 \\
 &= 100 + 836 \\
 A[6][2][5] &= 936
 \end{aligned}$$

Sparse matrices:

- Matrices with relatively high proportions of entries with zero are called sparse matrices.
- There are two general n-square sparse matrices.

1) Triangular matrix

- For a square 2-D array, if all the elements beneath(above) principal diagonal are zero, then the matrix is called as triangular matrix.
- If all elements beneath the principal diagonal are zero, then the matrix is upper triangular.
- While for a square matrix, if all the elements above the main diagonal are zero then matrix is upper triangular.

2) Tridiagonal Matrix

- For a square matrix, if all the elements are zero except the elements on the principal diagonal and all the elements immediately above and immediately below the principal diagonal, then the matrix is tridiagonal.



$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 5 & 2 & 0 & 0 \\ 6 & 7 & 3 & 0 \\ 8 & 9 & 10 & 4 \end{bmatrix}_{4 \times 4}$$

Lower triangular matrix

$$\begin{bmatrix} 1 & 5 & 8 & 10 \\ 0 & 2 & 6 & 9 \\ 0 & 0 & 3 & 7 \\ 0 & 0 & 0 & 4 \end{bmatrix}_{4 \times 4}$$



Upper triangular matrix

$$\begin{bmatrix} 1 & 5 & 0 & 0 \\ 8 & 2 & 6 & 0 \\ 0 & 9 & 3 & 7 \\ 0 & 0 & 10 & 4 \end{bmatrix}_{4 \times 4}$$

Tridiagonal matrix

Note:
A strictly lower or upper triangular matrix are those where the diagonal are also zero entries.

Triangular matrices:

1) Lower Triangular Matrix  $n \times n$, $a[i][j]$	
Range	$\frac{n(n+1)}{2}$
Row Major Order (RMO)	$b + \left[(j-1) + \frac{i(i-1)}{2} \right] * w$
Column Major Order (CMO)	$b + \left\{ (i-j) + \left[(j-1)n - \frac{(j-1)(j-2)}{2} \right] \right\} * w$
2) Upper Triangular Matrix  $n \times n$, $a[i][j]$	
Range	$\frac{n(n+1)}{2}$
Row Major Order (RMO)	$b + \left\{ (j-i) + \left[(i-1)n - \frac{(i-1)(i-2)}{2} \right] \right\} * w$



Column Major Order (CMO)	$b + \left[(i - 1) + \frac{j(j - 1)}{2} \right] * w$
3) Strictly Lower Triangular Matrix []nxn, a[i][j]	
Range	$\frac{n(n - 1)}{2}$
Row Major Order (RMO)	$b + \left[(j - 1) + \frac{(i - 1)(i - 2)}{2} \right] * w$
Column Major Order (CMO)	$b + \left[(i - 1) + \frac{(j - 1)(j - 2)}{2} \right] * w$
Tridiagonal matrix [] _{nxn} , a[i][j]	
Range	3n-2
Row Major Order (RMO)	b+ (2i+j-3) *w
Column Major Order (CMO)	b+ (i+2j-3) *w



Rack Your Brain

- 1) Let an array A[1.....6] [1.....10] be a 2-D array. The first element of the array is stored at location 100. Let each of the element occupy 8 bytes. Find the memory location of the element in the second row and fourth column when the array, (i) is stored in RMO and (ii) stored in CMO.
Hint : Find location of A[2][4]
- 2) Consider the following multi-dimensional array :
x[-5 : 5, 3 : 33], y[3 : 10, 1 : 15, 10 : 20]
- a) What would be the number of elements in x and y.
Hint : Range = (U.B -L.B + 1)
- b) Let the base address of array y be 400, and each element occupies 4 bytes of memory location. Than what is the address of memory location y[5, 10, 15] when y is stored in row major order and column major order.



Chapter Summary

- Data structure are the logical or mathematical model of a particular organization of data.
- Data structures are chosen such that they can represent the actual relationship of data in real world, while being simple enough that the data can be processed effectively.
- Arrays are simplest type of data structure.
- Arrays are of three types :
 - 1) Linear or 1-D arrays
 - 2) 2-D array
 - 3) n-D array or multidimensional arrays.
- A linear array or one-dimensional array is a list of finite number of similar data elements represented by a set of n consecutive numbers.
- An array A can be represented as :
 - 1) Subscript notation
 $A_1, A_2, A_3, \dots, A_n$
 - 2) Parenthesis notation
 $A(1), A(2), A(3) \dots A(n)$
 - 3) Bracket notation
 $A[1], A[2], A[3] \dots A[n]$

Here : the number N in $A[N]$ is the subscript.
- Arrays are by default stored in Row major order.
- The address of K^{th} element of 1-D array a is given as:
 $a[K] = b + [K - L.B] * w$
- The address of the element at i^{th} row and j^{th} column of a 2-D array is given as:
RMO : $A[i][j] = b + [(U_2 - L_2 + 1) (i - L_1) + (j - L_2)] * w$
CMO : $A[i][j] = b + [(i - L_1) + (j - L_2) (U_1 - L_1 + 1)] * w$
- The address of the element at $i^{\text{th}}, j^{\text{th}}$ and K^{th} dimension of a 3-D array is given as:
 $A[i][j][K] = b + \{(i - L_1) [(U_2 - L_2 + 1) (U_1 - L_1 + 1)]$
 $+ (j - L_2) [(U_1 - L_1 + 1)]$
 $+ (K - L_3)\} * w$