* Arrays, Records, Pointers.

Operations on linear structures

1. Traversal
2. Searching
3. Insertion
4. Deletion
5. Sorting
6. Merging

⇒ Linear Arrays

↳ List of finite no. 'n' of homogeneous data elements (elements of same type).

↳ Elements are referenced by 'index set' [1 to n]

↳ Array elements are stored in successive memory locations.

Length or size of array = UB − LB + 1

Example :- DATA = 247, 56, 429, 135, 87, 156.

AUTO [K] = no. of automobiles sold in year 'K'.

Representation in memory -

Computer Memory

Let - LOC [A[K]] = address of element A[K] of array A.

As elements are stored in continuous memory locations, no need to keep track of addresses of all elements. We can only track address of first element

Base (A) → base address of A.

then -

$$LOC(A[K]) = Base(A) + w(K - LB)$$

where -

w - no. of words per memory cell.

Ex: $LOC(AUTO[1932]) = 200$

$LOC(AUTO[1933]) = 204$

⋮

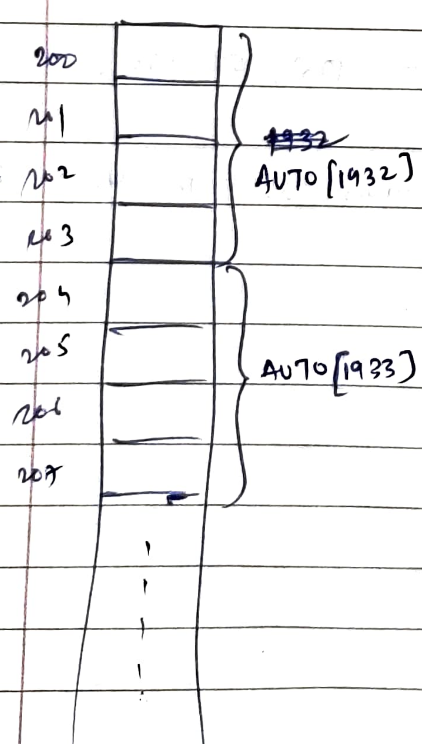
Address for $k = 1965$ -

$$LOC(AUTO[1965]) = Base(AUTO) + w(1965 - LB)$$

$$= 200 + 4(1965 - 1932)$$

$$= 200 + 4 \times 33$$

$$= \underline{332}$$



* Traversing the array

Algo :-

1 Repeat for K = LB to UB.
      Apply PROCESS to A[K].
   [End of loop]
2. Exit

Or

1. Set K := LB.
2. Repeat steps ③ & ④ while K <= UB.
3.   Apply PROCESS to A[K].
4.   K = K+1
. [End of while loop]
5. Exit.

Example :- Find no. of years when sales of automobiles is greater than 300 :
If AUTO[K] > 300 then set COUNT = COUNT + 1

or

Print year and no. of automobiles sold
Write K, AUTO[K].

* Insertion and Deletion

Insert :- Add element to array
Delete :- Remove element from array.

INSERT (A , N , K , ITEM). → Insert item at K^{th} position

1. Set J = N
2. Repeat 3 and 4 while J ≥ K.
3. Set A[J+1] = A[J]
4. Set J = J-1 ;

[End of loop].

5. Set A[K] = ITEM
6. Set N = N+1
7. Exit.

DELETE (A , N , K , ITEM)

1. Set ITEM = A[K]
2. Repeat for J = K to N-1

Set A[J] = A[J+1].

[End of loop]

3. Set N = N-1
4. Exit.

Note :- Array is not an efficient way of storing data when we have to frequently insert and delete items.

Assignment 1.

1) Write and algorithm for Bubble sort. Do its complexity analysis: Best, Average, Worst. Write a working example for 5 elements

2) Write an algorithm for Linear search. Do its complexity analysis: Best, Average, Worst

* Binary Search

BINARY (DATA , LB , UB , ITEM , LOC)

① Set BEG = LB END = UB MID = INT((BEG+END) /2)

② Repeat steps ③ & ④ while BEG ≤ END and DATA [MID] ≠ ITEM

③ If ITEM < DATA [MID] then:
Set END = MID -1
Else
set BEG = MID +1
[End of if]

④ Set MID = INT((BEG+END) /2)
[END of step 2 loop]

⑤ If DATA [MID] = ITEM then:
Set LOC = MID.
Else
Set LOC = NULL
[End of IF structure].

⑥ Exit.

Example:-

DATA = 11 , 22 , 30 , 33 , 40 , 44 , 55 , 60 , 66 , 77 , 80 , 88 , 99

ITEM = 40.

① BEG = 1 END = 13 MID = (1+13)/2 = 7 DATA(7) = 55

② 40 < 55 so END = MID - 1 = 6.
MID = INT ((1+6)/2) = 3 DATA [3] = 30

③ 40 > 30, so BEG = MID + 1 = 4.
MID = (4+6)/2 = 5 DATA [5] = 40.

We have found ITEM then LOC = MID = 5

Suppose ITEM = 85

	BEG	END	MID	DATA [MID]	Condition
1	1	13	7	55	55 < 85
2	8	13	10	77	77 < 85
3	11	13	12	88	88 > 85
4	11	11	11	80	80 < 85
5	12	11	→ BEG > END	→ out of loop.	

Complexity for Binary search

worst case when value is not present or present at extreme ends

Iterations	Size of Array
0	n
1	$n/2$
2	$n/4 = n/2^2$
⋮	
k	$n/2^k = 1$

$n = 2^k.$

$$\boxed{k = \log_2 n}$$

or form the recurrence equation.

$T(n) = T(n/2) + 1$

$T(n) = O(\log n)$

Example: DATA contains 1000000 elements

the $2^{10} = 1024 > 1000$

$2^{20} \geq (1000)^2 = 1000000$ } only 20 comparisons.

Multidimensional Array

$$
\begin{array}{c|cccc}
 & & \text{Columns} & & \\
\text{Rows} & A(1,1) & A(1,2) & A(1,3) & A(1,4) \\
 & A(2,1) & A(2,2) & A(2,3) & A(2,4) \\
 & A(3,1) & A(3,2) & A(3,3) & A(3,4)
\end{array}
$$

Two dimensional 3×4 Array.

$$\text{Length} = UB - LB + 1$$

Ex: In FORTRAN INTEGER NUMB (2:5, -3:1)

Length of 1st dimension: $5 - 2 + 1 = 4$

" " 2nd dimension: $1 - (-3) + 1 = 5$

Representation of 2-D Array in Memory.

Recall $LOC(A[K]) = Base(A) + w(K - LB)$

(a) Column-Major order

A		
	(1,1)	Colum 1
	(2,1)	
	(3,1)	
	(1,2)	Colum 2
	(2,2)	
	(3,2)	
	(1,3)	Colum 3
	(2,3)	
	(3,3)	
	(1,4)	Column 4
	(2,4)	
	(3,4)	

(b) Row Major order

A		
	(1,1)	Row 1
	(1,2)	
	(1,3)	
	(1,4)	
	(2,1)	Row 2
	(2,2)	
	(2,3)	
	(2,4)	
	(3,1)	Row 3
	(3,2)	
	(3,3)	
	(3,4)	

Column Major order.

$$LOC(A[J,K]) = Base(A) + w[M(K-1)+(J-1)]$$

Row Major order

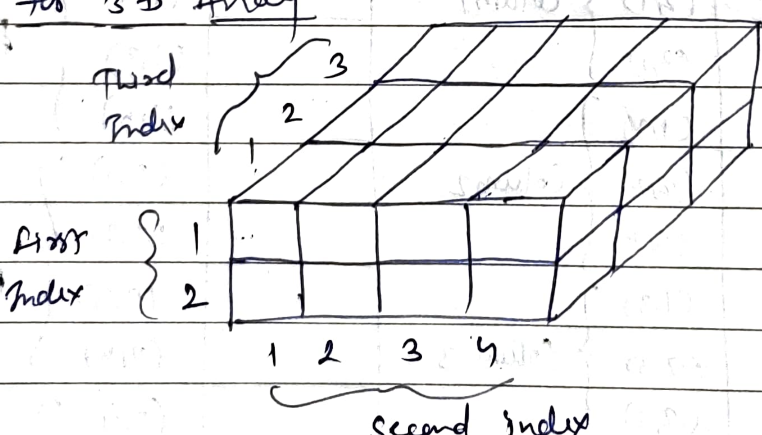
$$LOC(A[J,K]) = Base(A) + w[N(J-1)+(K-1)].$$

Example :-

Consider 25×4 Matrix SCORE.
Base (SCORE) = 200 and $w = 4$.
It is stored using row major order.
SCORE (12, 3) = ?

$$LOC(SCORE[12,3]) = 200 + 4[4(12-1)+(3-1)]$$
$$= 200 + 4[44+2]$$
$$= 384.$$

For 3D Array



$2 \times 4 \times 3$.
3-D Array

$L_1, L_2, L_3 \rightarrow$ dimensions

$E_1 = K_1 - LB$ $E_2 = K_2 - LB$ $E_3 = K_3 - LB$

$= K_1 - 1$ $= K_2 - 1$ $= K_3 - 1$