Arrays

→ An array is a collection of similar data elements.

Ex; to store names of all students in a class.

→ Array is a list of finite number n of homogeneous data elements such that the elements of the array are stored in successive memory locations.

→ Array is basically used to store relatively permanent collections of data, whereas if the size of the ~~structure~~ and data in the structure are constantly changing, then the array may not be as useful as the linked list.

→ Elements of the array are referenced respectively by an index set consiting of n consecutive nos.

→ ~~Ete~~ The length or no. of elements of an array can be obtained by the formula:-

$$\boxed{\text{Length} = \underset{\text{Upper bound}}{UB} - \underset{\text{Lower bound}}{LB} + 1}$$

When LB = 1, then UB = Length or size.

→ The elements of the array are denoted as:-

A[1] Subscript or index

Subscripted variable

Describes the position of any element in an array.

→ The array declaration must involve three items of information:

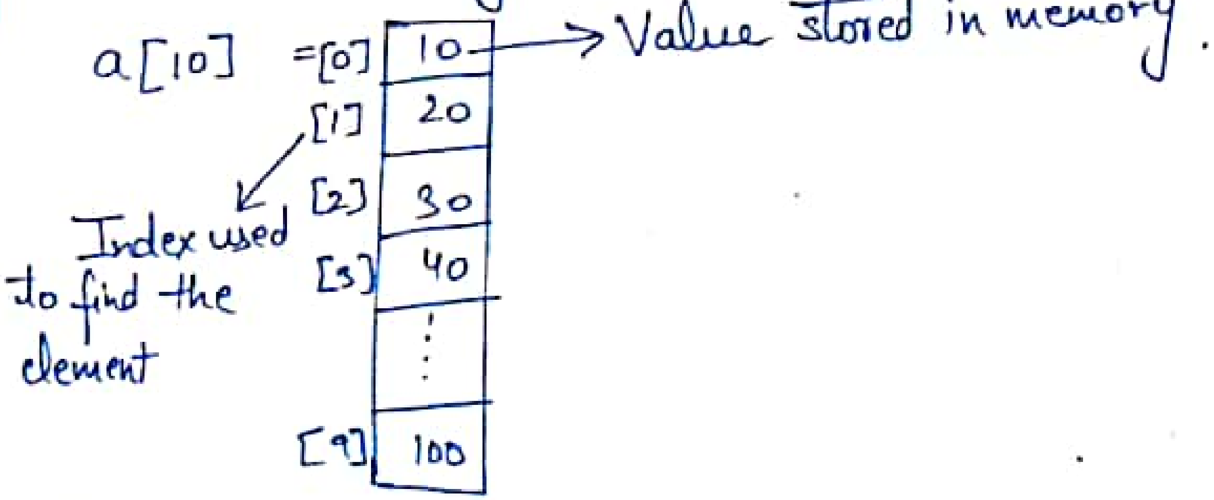
1) Name of the array
2) Data type of the array
3) Index set of the array

Eg., Int A[10].

→ One-Dimensional Array:-

Since, each element in the array is referenced by a single subscript, thus it is 1-D Array or linear arrays.

Eg., a[10] =



Array

⇒ Traversing in Array

LA, lower Bound = LB, Upper bound = UB, S → each element
(S = Process)

Algo:- i) Initialize Counter : I = LB
ii) loop (while I ≤ UB)
iii) Apply S to LA[I]
iv) I = I + 1
v) End loop
vi) Exit

Insertatloc(A, n, loc, ele)
// A is an array with n no of element and ele is element to be inserted at location loc

pos-4

1	2	3	4	5	6	7	8	9	10
10	20	30	40	50	60	70			

I = 7

1. I = n
2. Repeat steps 3 to 4 while (I ≥ LOC)
3. Set A[I+1] = A[I]
4. Set I = I-1
[End of loop]
5. Set A[LOC] = ele
6. Set N = N+1
7. Exit.

Algorithm to delete an element at given location
Delete at LOC(A, n, loc)...
// A is an array with n no of element and loc is location

10	20	30	40	15	2	8
[1]	[2]	[3]	[4]	[5]	[6]	[7]

1. Set I = LOC
2. Repeat steps 3 to 4 while (I < N)
3. Set A[I] = A[I+1]
4. I = I+1
[End of loop]
5. Set N = N-1
6. Exit.

29/08/16.

Representation of Linear Array in memory -

marks[] = {99, 67, 78, 56, 88, 90, 34, 85}
marks[4] = ? BA = 1000
marks[4] = 1000 + 2(4-0)
= 1008

for 1-D array. A[K] = Base(A) + w(K - Lower Bound)

Memory Rep

Consider an Array A Base address = 2000 and its index start from 1932 No. of words required are 4 Find the address of 1965

A[1965] = 2000 Base add. + w(K - lower bound)

= 2000 + 4(1965 - 1932)

= 2000 + 4 × 33

= 2000 + 132 ⇒ 2132

2D Array Representation in memory

	1	2	3	4
1	A[1,1]	A[1,2]	A[1,2]	A[1,4]
2	A[2,1]	A[2,2]	A[2,3]	A[2,4]
3	A[3,1]	A[3,2]	A[3,3]	A[3,4]

There are two ways of representing 2-D array

1. Row Major Order Implementation :

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

$$A[J,K] = \text{Base address of } A + w\big(\underset{\text{length of column}}{N}(\underset{\text{for row}}{J - \text{Lower Bound}}) + (\underset{\text{for col}}{K - LB})\big)$$

2. Column Major Order Implementation :

A[1,1]
A[2,1]
A[3,1]
A[1,2]
A[2,2]
A[3,2]
⋮
A[3,4]

By default LB = 1

$$A[J,K] = \text{Base address}(A) + w[\underset{\text{length of row}}{M}(\underset{\text{for col.}}{K - LB}) + (\underset{\text{for row}}{J - LB})]$$

eg. A [2...6, 3...10]

Suppose a 2D Array A is declared as $A[-2:2, 2:6]$

a) words per cell = 4. Base address = 200

a) Find out length of each dimension and no. of elements in array

b) Find the location of $A[1,2]$

a) length of row = $2-(-2)+1 = 5$

length of column = $6-2+1 = 5$

No of elements = $5 \times 5 \Rightarrow 25$

b) Row Major -

$A[1,2] = 200 + 4(5(1-(-2)) + (2-(+2)))$

$200 + 4[15]$

$200 + 60 = 260$

Column Major -

$A[1,2] = 200 + 4[5(2-2) + (1-(-2))]$

$200 + 4[3] \Rightarrow 212$

Consider a 2D Array $A[25][4]$ B.A = 200 and it requires 4 words memory cell. Find location of $A[12][3]$ if the array is stored row major

$A[12][3] = 200 + 4[4(12-1) + (3-1)]$ (Assume array index from 1)

$200 + 4[4 \times 11 + 2]$

$200 + 4[46] = 200 + 184 = 384$

Array index = 0 then $A[12][3] = 404$

General Multi Dimensional Array :

An n dimensional $(m_1 \times m_2 \dots m_n)$ Array A is a collection of elements of $(m_1 \cdot m_2 \cdot m_3 \dots m_n)$ data elements in which each element is specified by a list of n integers such as $k_1, k_2 \dots k_n$ are called subscript with property eg. $A[2][3][3]$

$1 \leq K_3 \leq m_3$ $\quad 1 \leq K_1 \leq m_2$ $\quad 1 \leq K_2 \leq m_2$

$5-1+1$ $\quad$ $4-1+1$

Length of Row $= 5$ $\quad$ length of col $= 4$

Q. Calculate the address of X[4,3] in a 2D-array X(1...5, 1...4) stored in a row-major order in the main memory. Assume the base address to be 1000 and that each element requires 4 words of storage.

$$X[4,3] = 4([\overset{I}{(4-1)}\overset{N}{4}+\overset{J}{(3-1)}]+1000$$

$$= 1056$$

Multidimensional Array

$E_i = K_i -$ Lower Bound $\quad$ $L_i = UB - LB + 1$

Column Major Order:

$$Loc(A[K_1, K_2, \ldots K_n]) \Rightarrow$$

$$\Rightarrow Base(A) + w[(((\ldots(E_N L_{N-1} + E_{N-1})L_{N-2} + \ldots + E_3)L_2 + E_2)L_1 + E_1]$$

for $n=1 \Rightarrow Base(A) + w[E_1]$

$n=2 \Rightarrow Base(A) + w[E_2 L_1 + E_1)$

$n=3 \Rightarrow Base(A) + w((E_3 L_2 + E_2)L_1 + E_1)$

$n=4 \Rightarrow Base(A) + w[((E_4 L_3 + E_3)L_2 + E_2)L_1 + E_1]$

Row Major Order

$$Loc(A[K_1, K_2, \ldots K_n]) \Rightarrow Base(A) + w[(\ldots((E_1 L_2 + E_2)L_3 + E_3)L_4 + \ldots + E_{N-1})L_N + E_N]$$

1D Address Calculation Numerical

1. $A[-15 \ldots 64]$ is stored in computer memory whose base address is 459. word size is 2 byte

a) How many no. of element are there.
b) Total memory size (size of array X word size)
c) Find memory location of $A[10]$
d) Which element is located in memory address. 589.

a. Size of array $= 64-(-15)+1$ UB-LB+1
$= 80$

b) $80 \times 2 = 160$ byte

c) $A[10] = B.A + w(K - LB)$
$= 459 + 2(10-(-15))$
$= 459 + 2(25)$
$= 459 + 50$
$= 509$

d) $589 = 459 + 2(K-(-15))$
$589 = 459 + 2(K+15)$
$589 = 459 + 2K + 30$
$589 - 489 = 2K$
$100 = 2K$
$K = 50$

$$\begin{array}{r} 459 \\ 30 \\ \hline 489 \end{array}$$

Multi-dimensional Array Numerical

1. Suppose A is a 3D array given as : $A(1:9, -4:1, 5:10)$ The array stores in memory in Row-major order and base address = 400 and w = 2 words per memory cell. Then find out the address of Element $A(5,-1,8)$ row-major wise.

Sol:

Row-Major:

$$LOC\ A[i][j][k] = Base(A) + w[(E_1 L_2 + E_2) L_3 + E_3]$$

$L_1 = 9 - 1 + 1 = 9$ $\quad$ $E_1 = 5 - 1 = 4$

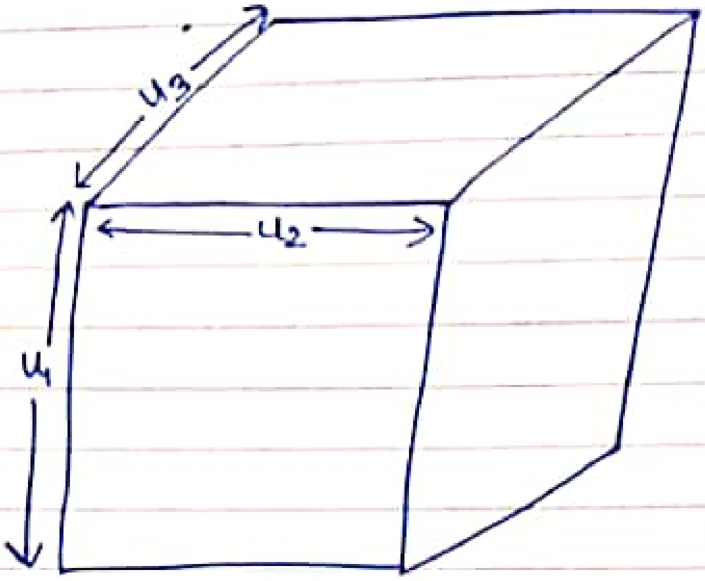
$L_2 = 1 - (-4) + 1 = 6$ $\quad$ $E_2 = -1 - (-4) = 3$

$L_3 = 10 - 5 + 1 = 6$ $\quad$ $E_3 = 8 - 5 = 3$

$$LOC\ A[5][-1][8] = 400 + 2[(4 \times 6 + 3)6 + 3]$$
$$= 730$$

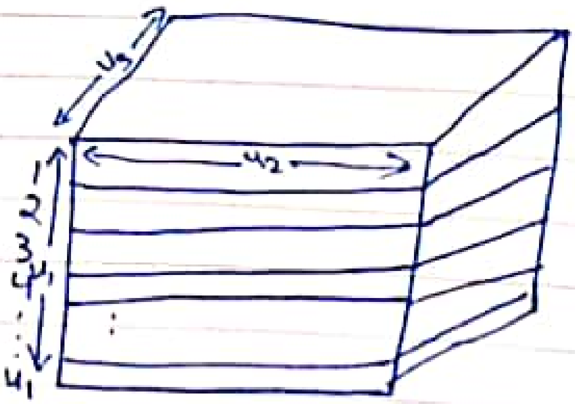
Scanned with CamScanner

Index Formula Computation
3-D Array Row Major Order

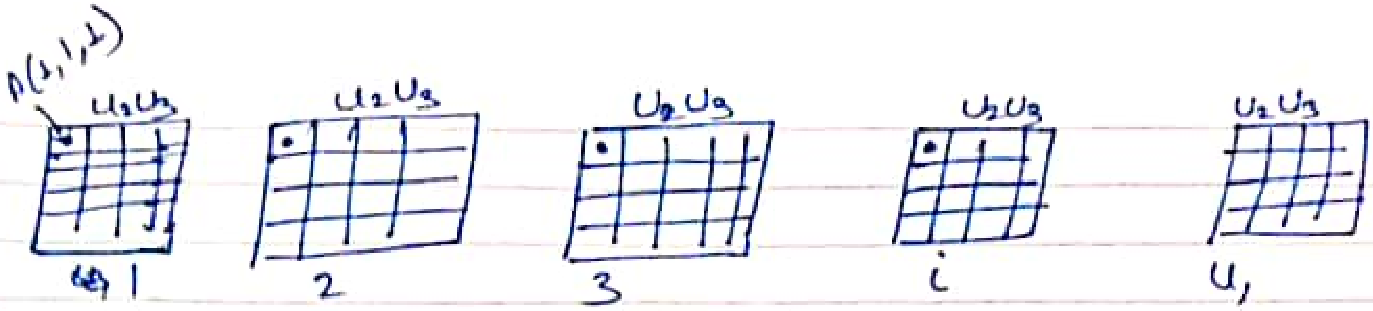


Let us say there is first dimension representing u_1. Second dimension u_2. Third dimension u_3.
We can imagine it as a cuboid.
Array is:
$A[L_1:U_1, L_2:U_2, L_3:U_3]$

For the assumptions let's say the first index is 1. So, $A[1:U_1, 1:U_2, 1:U_3]$ and every element is requiring one byte for the storage.
Now, if I say this is a 3D array and we break it in various pieces or the slices.



So the total no. of slices can be cut upto u_1.



Every slice is of size $u_2 \times u_3$

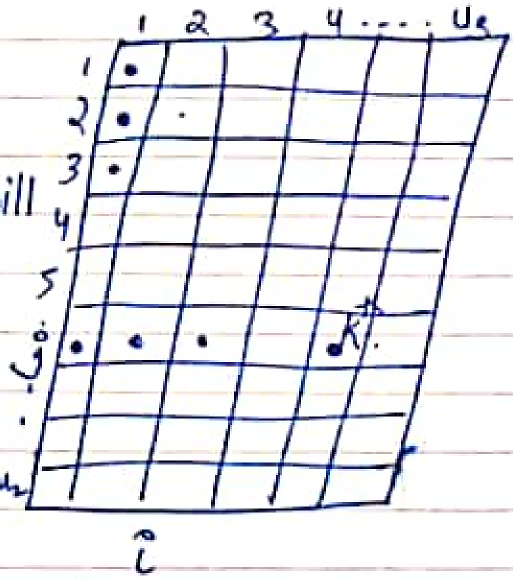
So, if I know the address of 1st element that is,

$A[1,1,1] = \alpha$, then what will be address of

$A[2,1,1] = \alpha + u_2 \times u_3$ (As u can only come to the storage of this element after storing 1st 2D slice)

Similarly, $A[3,1,1] = \alpha + 2 u_2 u_3$
Similarly, $A[i,1,1] = \alpha + (i-1) u_2 u_3$

Now let us expand i array:
So, we already know the address of 1st element, what will be the address of 1st element of 2nd row 6. It will be address of 1st row 1st element + u_3



$A[i,2,1] = \alpha + (i-1) u_2 u_3 + u_3$
$A[i,3,1] = \alpha + (i-1) u_2 u_3 + 2 u_3$
Similarly, address of 1st element of jth row
$A[i,j,1] = \alpha + (i-1) u_2 u_3 + (j-1) u_3$
Similarly $A[i,j,2] = \alpha + (i-1) u_2 u_3 + (j-1) u_3 + 1$
$A[i,j,3] = \alpha + (i-1) u_2 u_3 + (j-1) u_3 + 2$
Similarly for kth element of jth row
$A[i,j,k] = \alpha + (i-1) u_2 u_3 + (j-1) u_3 + (k-1)$ — (1) this is a formula with assumptions.

Noo, if I try to remove assumptions, then i is

$A[i,j,k] = \alpha + (i-1)u_2u_3 + (j-1)u_3 + (k-1)$

on 1st dimension side, j for 2nd dimension & k for 3rd dimension and every element is acquiring n byte for storage.

So, removing the assumptions:

$A[i,j,k] = \alpha + [(i-1)u_2u_3 + (j-1)u_3 + (k-1)]n$

i will be replaced by $i-L_1+1$, j by $j-L_2+1$ and k by $k-L_3+1$

$$A[i,j,k] = \alpha + [(i-L_1)(u_2-L_2+1)(u_3-L_3+1) + (j-L_2)(u_3-L_3+1) + (k-L_3)] \times n \quad (2)$$

So, then replacing

$i-L_1 = E_1$

$j-L_2 = E_2$

$k-L_3 = E_3$

$\alpha = B.A$

$u_2-L_2+1 = L_2$

$u_3-L_3+1 = L_3$

$w n = L_4$

$= B.A + [((E_1 L_2 + E_2)L_3 + E_3)L_4 \, \omega + E_4]\omega$

Scanned with CamScanner