Address Calculation in 1-D & 2-D array

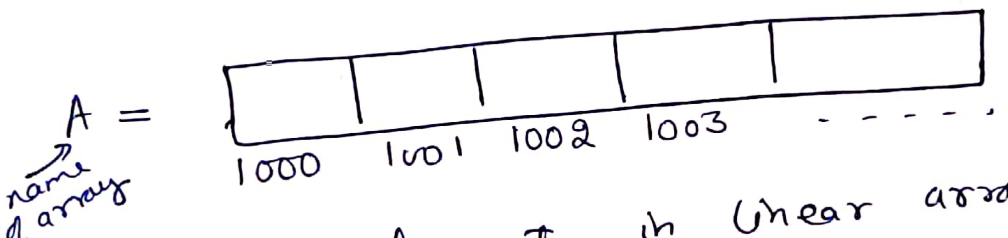
The number of elements is called the size of the array or length of the array.

No. of elements in a array can be obtained by

$\Rightarrow UB - LB + 1$

upper bound, lower bound.

Representation of Linear Array (1-D) in memory



As ~~linear~~ elements in linear array are stored in successive memory cells. Therefore computer does not need to keep track of the address of every element, But needs to keep track of only the address of first element & that is denoted by B.A → Base address.

Using B.A., computer calculates the address of any element of linear array. by using the formula:→

$$A[i] = B.A + W(i - LB)$$

where :-

$A[i] = B.A + W(i - LB)$

Where

$B.A \rightarrow$ Base address of linear array

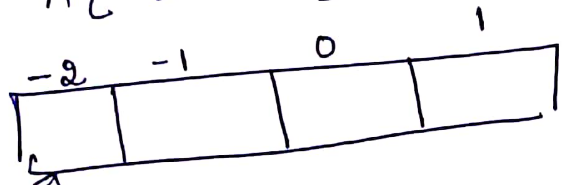
$W \rightarrow$ no. of words per memory cell

[eg In window OS int take 2 byte where as in LINUX int take 4 byte]

$i \rightarrow$ Given any subscript [position]

$LB \rightarrow$ lower bound of the array.

eg $A[-2 \ldots 1]$



say the BA is 200

Say we want to compute for $A[0]$ & word size is 2byte

$A[0] = 200 + 2(0 - (-2))$

200 → BA, 2 → W, 0 → position for which we are computing, (-2) → LB.

$A[0] = 200 + 4 \Rightarrow \underline{\underline{204}}$.

Q An array A[-15 .... 64] is stored in Computer memory whose Base address is 459. word size is 2 byte

(a) How many no. of elements are there.

Ans size of array = UB − LB + 1

$\Rightarrow 64 - (-15) + 1$

$\Rightarrow 80$ elements

(b) Total memory size.??

Ans ~~size~~ Total memory size = size of ~~memr~~ array * word size

$\Rightarrow 80 * 2$

$\Rightarrow 160$ bytes.

(c) Find the location of A[10]

Ans $A[i] = BA + W(i - LB)$

$A[10] = 459 + 2(10 - (-15))$

$\Rightarrow 459 + 2(10+15)$

$\Rightarrow 509$

(d) which element is located in memory address 589 ??

Ans In this we have to find $\boxed{i = ??}$

$589 = 459 + 2(i + 15)$ → we have to find i

$130 = 2(i+15)$

$\boxed{i = 50}$ so we locate at A[50].

2-D array :-

$(\overset{\text{rows}}{m} \times \overset{\text{column}}{n} \text{ array})$

2-D arrays are represented in memory with help of 1-D array. There are two ways to represent 2-D array in memory

- Row - major order
- Column - major order

In row-major :- Elements are stored row-wise. Rows are listed on the basis of columns.

Consider a 2-D array.

$$a = \begin{array}{c} \begin{pmatrix} a_{00} & a_{01} & a_{02} \\ a_{10} & a_{11} & a_{12} \\ a_{20} & a_{21} & a_{22} \end{pmatrix} \end{array} \begin{array}{l} \leftarrow \text{row } 0 \\ \leftarrow \text{row } 1 \\ \leftarrow \text{row } 2 \end{array}$$

$$\underbrace{a_{00} \mid a_{01} \mid a_{02}}_{\text{row } 0} \mid \underbrace{a_{10} \mid a_{11} \mid a_{12}}_{\text{row } 1} \mid \underbrace{a_{20} \mid a_{21} \mid a_{22}}_{\text{row } 2}$$

In column-major :- Elements are stored column wise. Column are listed on basis of rows.

Consider a 2-D array

$$a = \begin{pmatrix} a_{00} & a_{01} & a_{02} \\ a_{10} & a_{11} & a_{12} \\ a_{20} & a_{21} & a_{22} \end{pmatrix}$$

$$\begin{array}{ccc} \text{col } 0 & \text{col } 1 & \text{col } 2 \end{array}$$

$$\underbrace{\boxed{a_{00}} \boxed{a_{10}} \boxed{a_{20}}}_{\text{Col } 0} \underbrace{\boxed{a_{01}} \boxed{a_{11}} \boxed{a_{21}}}_{\text{Col } 1} \underbrace{\boxed{a_{02}} \boxed{a_{12}} \boxed{a_{22}}}_{\text{Col } 2.}$$

i.e elements are stored column by column.

The computer keeps track of Base address (B.A) of the first element of the given array and computes the address of any element with help of the formula.

Row - Major formula :-

$$A[i, j] = B.A + w\left(n(i - L_1) + (j - L_2)\right)$$

where

$B.A \rightarrow$ Base address of an array

$w \rightarrow$ word size

$n \rightarrow$ no. of columns

particular row & column $\Rightarrow$ $[i$ & $j \rightarrow$ positions for which calculation is being done.

$L_1 \rightarrow$ lower bound of row

$L_2 \rightarrow$ lower bound of column.

We represent matrix $A[\underbrace{m}_{\text{rows}} \times \underbrace{n}_{\text{column}}]$

Formula for Column-major

$$A[i,j] = B \cdot A + w((i-L_1) + m(j-L_2))$$

position for which we are computing

<u>where :-</u>

$BA \rightarrow$ Base address of given array

$w \rightarrow$ word size

$i \& j \rightarrow$ position for which calculation is being made

$L_1 \rightarrow$ lower bound of rows

$L_2 \rightarrow$ lower bound of columns.

$m \rightarrow$ no. of rows.

How to identify the lower & upper bound of rows & column

eg $A[\underbrace{10:15}_{\text{It represent rows}}, \underbrace{-2:2}_{\text{It represent columns.}}]$

(10 → LB, 15 → UB; −2 → LB, 2 → UB)

no. of rows $\Rightarrow$ UB − LB + 1

$\Rightarrow 15 - 10 + 1 \Rightarrow 6$ rows.

no. of columns $\Rightarrow$ UB − LB + 1

$\Rightarrow 2 - (-2) + 1$

$\Rightarrow 5$ columns.

Q Assume an Array Arr [4 : 9 , -1 : 3]
2 byte storage for each element , Base address of the array is 100 , what will be the address of Arr [6][2] = ?? If the array is stored using row-major .

Ans: $A[i][j] = B.A. + W(n(i - L_1) + (j - L_2))$

$A[6][2] = 100 + 2(n(6-4) + (2-(-1)))$

this no. of column can be computed by
UB - LB + 1
$3 - (-1) + 1$
$\Rightarrow 5.$

$A[6][2] = 100 + 2(5(6-4) + (2-(-1)))$

$\Rightarrow 100 + 2(10+3)$

$\Rightarrow 126$ Ans.

Q A 2-D array defined as A [-4 : 6] [-2 : 12] require 2 byte of space for each element . If the array is stored in row major order with the address A [4] [8] as 4142 . Compute the address of A [0] [0]

Solution :-

Formula for Row major

$A[i][j] = B.A + W[\,n(i - L_1) + (j - L_2)\,]$

What is given to us :-

$W = 2$

$L_1 = -4$

$L_2 = -2$

no. of rows $\Rightarrow$ $UB - LB + 1$

$6 - (-4) + 1 \Rightarrow 11$

no. of columns $\Rightarrow$ $12 - (-2) + 1 \Rightarrow 15$

Address of $A[4][8] = 4142$

Now putting these values in the formula to obtain the B.A of the given array

$4142 = B.A + 2[\,15(4 - (-4)) + (8 - (-2))\,]$

$4142 = B.A + 260$

$B.A \Rightarrow 4142 - 260 \Rightarrow 3882$

Now we compute the address of $A[0][0]$

$A[0][0] = 3882 + 2[\,15(0 - (-4)) + (0 - (-2))\,]$

$\Rightarrow 3882 + 124$

$\Rightarrow 4006$