

2D Array

| | 0 | 1 | 2 | 3 | 4 | |
|---|-----|-----|-----|-----|-----|---|
| 0 | 10 | 20 | 30 | 40 | 50 | ✓ |
| 1 | 60 | 70 | 80 | 90 | 100 | ✓ |
| 2 | 110 | 120 | 130 | 140 | 150 | ✓ |
| 3 | 160 | 170 | 180 | 190 | 200 | ✓ |

array \Rightarrow { 0, 1, 2, 3 }

Row Num \uparrow

array[3][4] \downarrow Column Num

4x5

array[0][0] \rightarrow Base address \rightarrow 1000

Row major form

$n_r = 4$
 $n_c = 5$

Elements need to skip

$$\text{Loc}(\text{array}[i][j]) \Rightarrow 1000 + \left[(i - 0) * 5 + (j - 0) \right] * 2$$

$$\Rightarrow 1000 + (15 + 4) * 2$$

$$\Rightarrow 1000 + 38$$

$$\Rightarrow \underline{\underline{1038}}$$

In Row major form:
For searching 200 we need to skip 3 rows where each row is having 5 elements and in the last row have to skip 4 elements

Row major form

$$\text{Loc}(\text{array}[i][j]) \Rightarrow \text{Base address} + \left[(i - \text{LB}_r) * n_c + (j - \text{LB}_c) \right] * \text{size of each element}$$

lower bound of row which is by default=0 no. of columns

lower bound of column which is by default=0

Column major form

$$\text{Loc}(\text{array}[i][j]) \Rightarrow \text{Base address} + \left[(j - \text{LB}_c) * n_r + (i - \text{LB}_r) \right] * \text{size of each element}$$

Going forward we are going to discuss different application of an array like searching and sorting algorithm, divide and conquer approach.

size of each element