Array.

Scalar variable → int x = 10;
Vector variable → int A[5];

Declaration with initialization → int A[5] = {1, 2}
(size)

1	2			

- Same data type
- Contiginous memory location.
- To print it %u is used
  (used for unsigned integer as Addresses are not signed.)

Array in Stack → int a[10]

Allocating an Array in heap → int * p = new int[5]
or int * p = (int *) malloc (5 * (sizeof(int)))

Deallocating an array in heap → delete [] p;
free(p);

Accessing array in heap = p[0] = 5

Static Array	Dynamic array
→ Memory alocated at compile time	Memory allocatted at run time
→ size is fixed becz memory for array is contigeous	size not fixed (i.e can be fixed)
→ stored in stack	stored in heap

2D-array.

↓ In stack

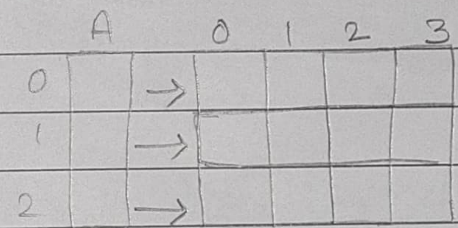
① int A [3] [4] ; → Method 1 to make a 2D array.

no of Row no of column

Method 2

int * A[3];

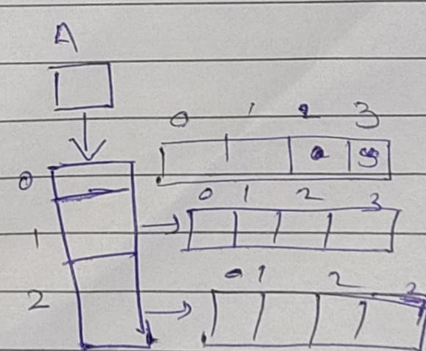
A[0] = new int [4]
A[1] = new int [4]
A[2] = new int [4]



Method 3

int * * A;
A = new int*[3]

A[0] = new int [4];
A[1] = new int [4];
A[2] = new int [4];



Row major mapping.

→ How to convert any index to address in array

A[m][n]

$Add(A[i]) = L_0 + i * w$

L_0 ↑ Base Address; i ↑ index number; w ↖ Size of Data type.

→ If index start from 1 → $L_0 + (i-1) * w$

→ For 2D-array - How to convert any index to address in array.

~~$Add(A[i][j]) = L_0 + [i]$~~

→ $Add(A[i][j]) = L_0 + (i * n + j) * w$ ← Size of Data type

L_0 ↗ Base address; i ↗ index no. of row; n ↑ no of columns; j ↖ index no of column

If index starts from 1 = $L_0 + ((i-1) * n + (j-1)) * w$

Column major mapping.

A [M] [n]

A	a_{00}	a_{10}	a_{20}	a_{01}	a_{11}	a_{21}

$$A = \begin{array}{c|cc} & 0 & 1 \\ \hline 0 & a_{00} & a_{01} \\ 1 & a_{10} & a_{11} \\ 2 & a_{20} & a_{21} \end{array}$$

A[2][1]

$$A[i][j] = L_0 + (((j \times m) + i) * w$$

L_0 = Base address, m = no of Rows, w = Size of Datatype

Row / column Major array using formula for n-dimension.

Type A[d_1][d_2][d_3][d_4]

Row major

~~$A[i_1][i_2][i_3][i_4] = L_0 + (i_1 + i_2 * d_2 * d_3 * d_4 + i_3 * d_3 * d_4$ + i~~

$$A[i_1][i_2][i_3][i_4] = L_0 + (i_1 \times d_2 \times d_3 \times d_4 + i_2 \times d_3 \times d_4 + i_3 \times d_4 + i_4) * w$$

Column major

$$A[i_1][i_2][i_3][i_4] = L_0 + (i_4 \times d_3 \times d_2 \times d_1 + i_3 \times d_2 \times d_1 + i_2 \times d_1 + i_1) \times w$$

Here Multiplication $= (n-1) + (n-2) + \dots 1$

$$= \frac{n(n-1)}{2}$$

$\therefore$ Time complexity $= O(n^2)$

Date ....
Page ....

~~Row major = L_0,~~

By Honor's Rule

Row major = $L_0 + [i_1 * d_2 \times d_3 * d_4 + i_2 * d_3 * d_4 + i_3 * d_4 + i_4] * w$.

$= \cancel{L_0 + d_4 * [i_3}$

$= \cancel{L_0 4}$

$= i_4 + i_3 \times d_4 + i_2 * d_4 * d_3 + i_1 * d_2 * d_3 \times d_4$

$= i_4 + d_4 * [i_3 + i_2 * d_3 + i_1 * d_3 * d_2]$

$= i_4 + d_4 * [i_3 + d_3 * [i_2 + i_1 * d_2]$

$\therefore$ Here Time complexity $\rightarrow O(n)$

3-D array:

int A[l][m][n];

Row-major

~~Row~~ Add(A[i][j][k]) $= L_0 + (i \times m \times n + \cancel{j} \times n \} + k)_w$

Add(A[i][j][k] $= L_0 + (i * m * n + j * n + k) * \omega$

Column major

A[i][j][k] $= L_0 + (k * m * l + j * l + i] * \omega$

* Quiz

x[4][3] = {{1, 2, 3}, {4, 5, 6}, {7, 8, 9}, {10, 11, 12}}

x+3 ⇐ 3rd row address

*(x+3) ← 3rd row address

*(x+2)+3 ← ~~2nd row 3rd column~~ 3rd row address