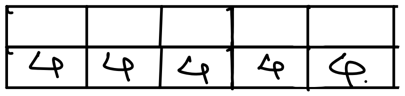
Consider array 5 of integers.

int a[5];

(1) a is ____________________ a :

(2) a is array 5 of ______________ a :

(3) a is array 5 of integers __________ a :

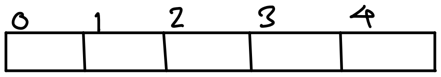


```
struct Date{
   int day;
   int month;
   int year;
};
```

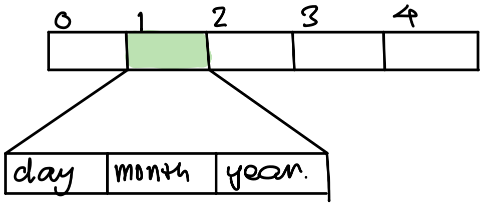
struct Date dt_arr[5];

1) dt_arr is: dt_arr :

2) dt_arr is array 5 of: dt_arr :



3) dt_arr is array 5 of struct Date:

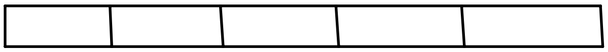


---

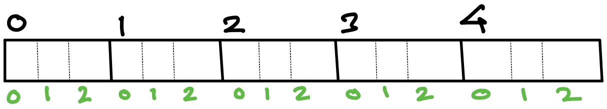
3) int a[5][3];

1) a is : a :

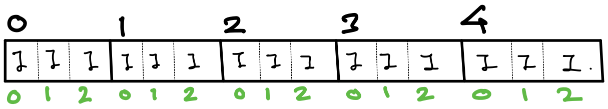
2) a is array 5 of : a :



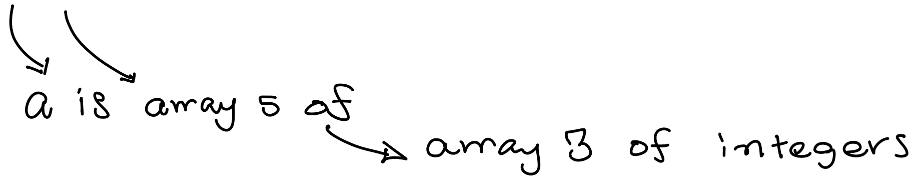
3) a is array 5 of array 3 of



4) a is array of 5 of array 3 of int:



int a[5] [3]



a is array 5 of

array 3 of integers

4) int a[3][5][4];

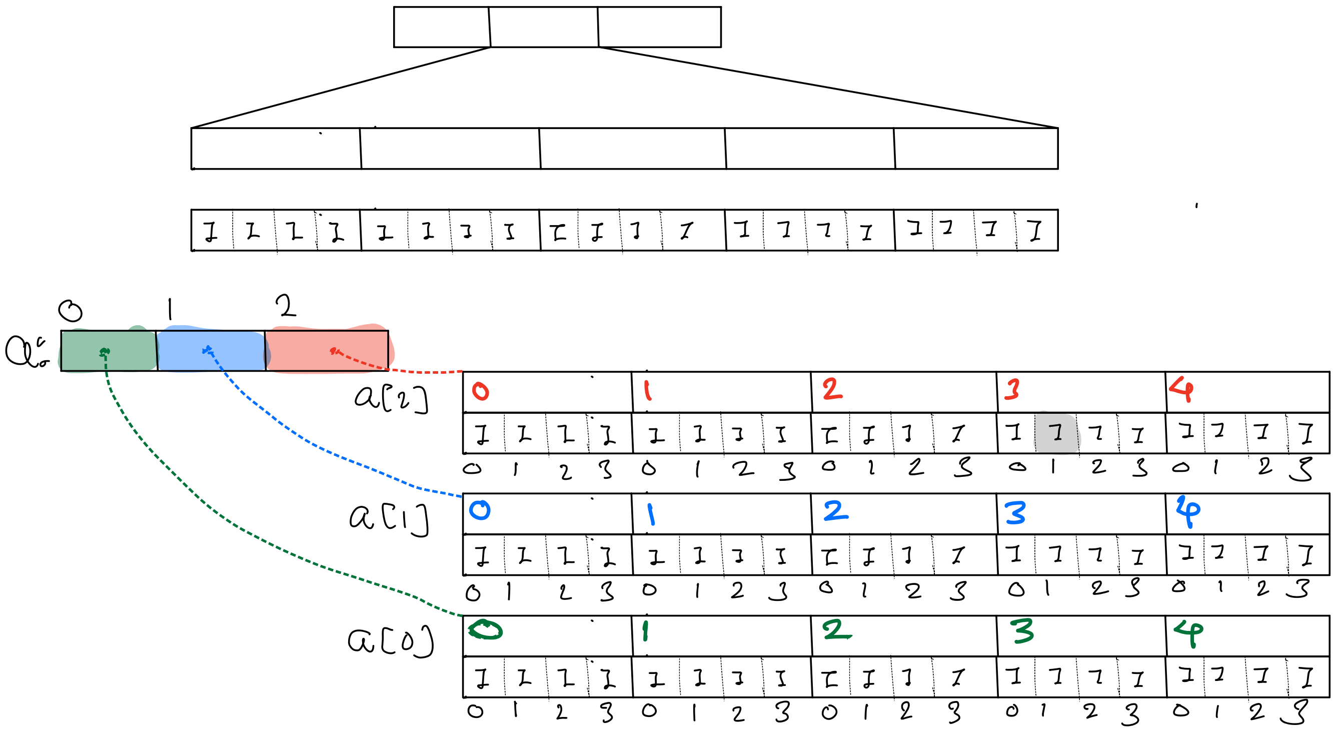
1) a is

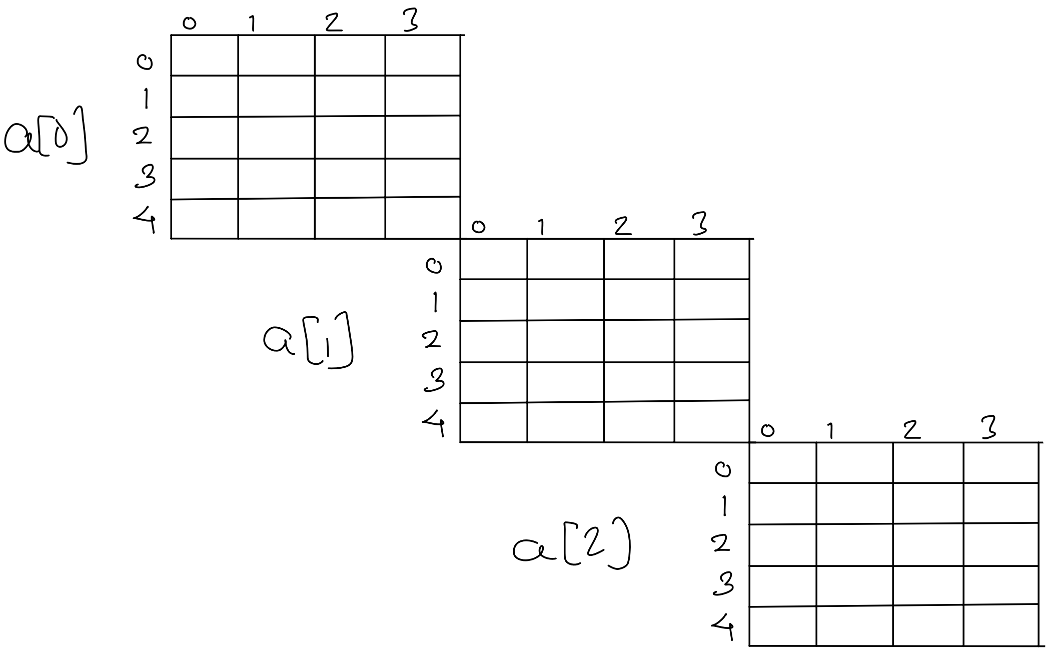
2) a is array 3 of

3) a is array 3 of array 5 of

4) a is array 3 of array 5 of array 4 of

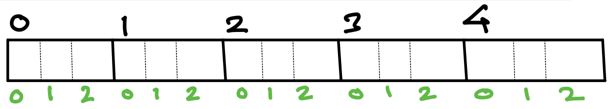
5) a is array 3 of array 5 of array 4 of int.



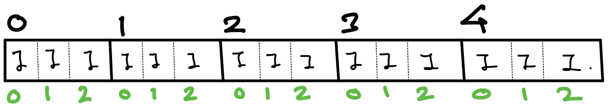


1) a is : a :

2) a is array 5 of



3) a is array 5 of array 3 of int



int a[5][3]

type(a) (in the context of array access) : int   [5][3]

type(a[i]) : int   [3] where i=0,1,2,3,4

type(a[i][j]) : int   where i=0,1,2,3,4 and j=0,1,2

a is array: therefore, [index] can be applied on it/.

Important fact: as 'a' is array, the expression a[i] is also an array.

a[i][j][k]

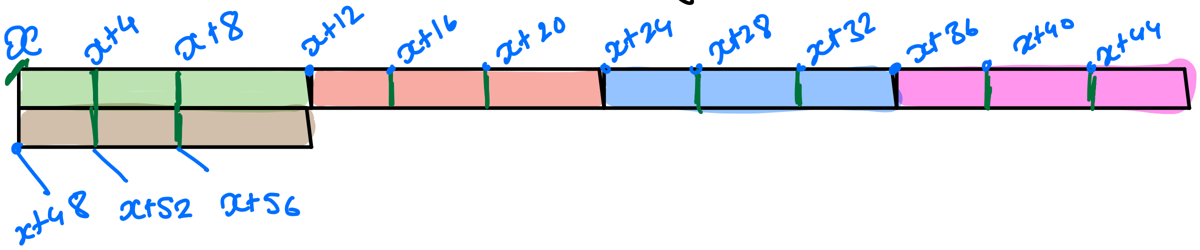
int a1[5]; : type(a1) :   int   [5]

type(a1[i]) : int   where i=0,1,2,3,4

```
a1[i][j]

type(a1[i]) == int, and int is not compatible with [index] operator,
a1[i][j] becomes invalid.

int a2[5][3];

type(a2[i][j]) == int where i=0,1,2,3,4 and j=0,1,2

a2[i][j][k]

type(a2[i][j]) == int and it is not compatible with [index]. therefore invalid

a2 is an array.
a2[index_1]

a2[index_1] is also an array as long as index_1=0,1,2,3,4
a2[index_1][index_2] as long as index_1=0,1,2,3,4 and index_2=0,1,2

int a[N];    a[i]

int a[N1][N2];   a[i][j]

int a[N1][N2][N3]; a[i][j][k];
```

---

int a[5][3] : a is array 5 of.



Compiler : int a[5][3];

$3 \times 5 \times \text{sizeof(int)} = 3 \times 5 \times 4 = 15 \times 4 = 60$

&a[0] = x

&a[1] = $x+12$

&a[2] = $x+24$

&a[3] = $x+36$

&a[4] = $x+48$.

int a[5][3]

type(a) = int [5][3]

type(a[i]) = int [3] whene $i=0,1,2,3,4$.

a[i] = Base Address of a
+ i * sizeof(typeof(a[i]))

= Base Address of a
+ i * sizeof(int [3]).

= Base Address of a
+ i * 3 * sizeof(int)

= Base Address of a
+ i * 3 * 4.

= Base Address of a
+ i * 12

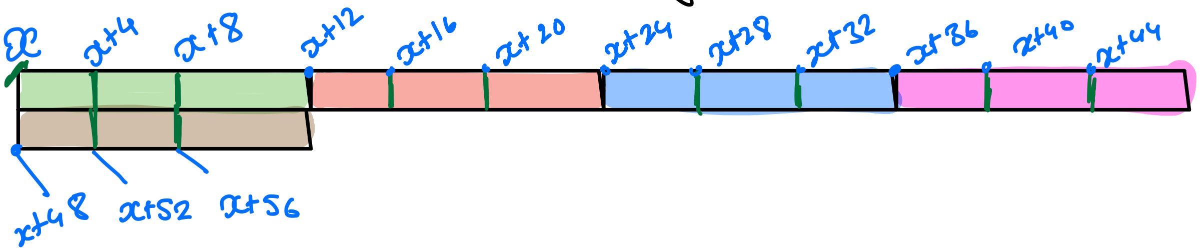
a[i] = x + i * 12

$i=0,1,2,3,4$, x+0 * 12, x+1*12, x+2* 12, x+ 3*12, x+4*12

x, $x+12$, $x+24$, $x+36$, $x+48$

int a[5][3] : a is am 5 of.

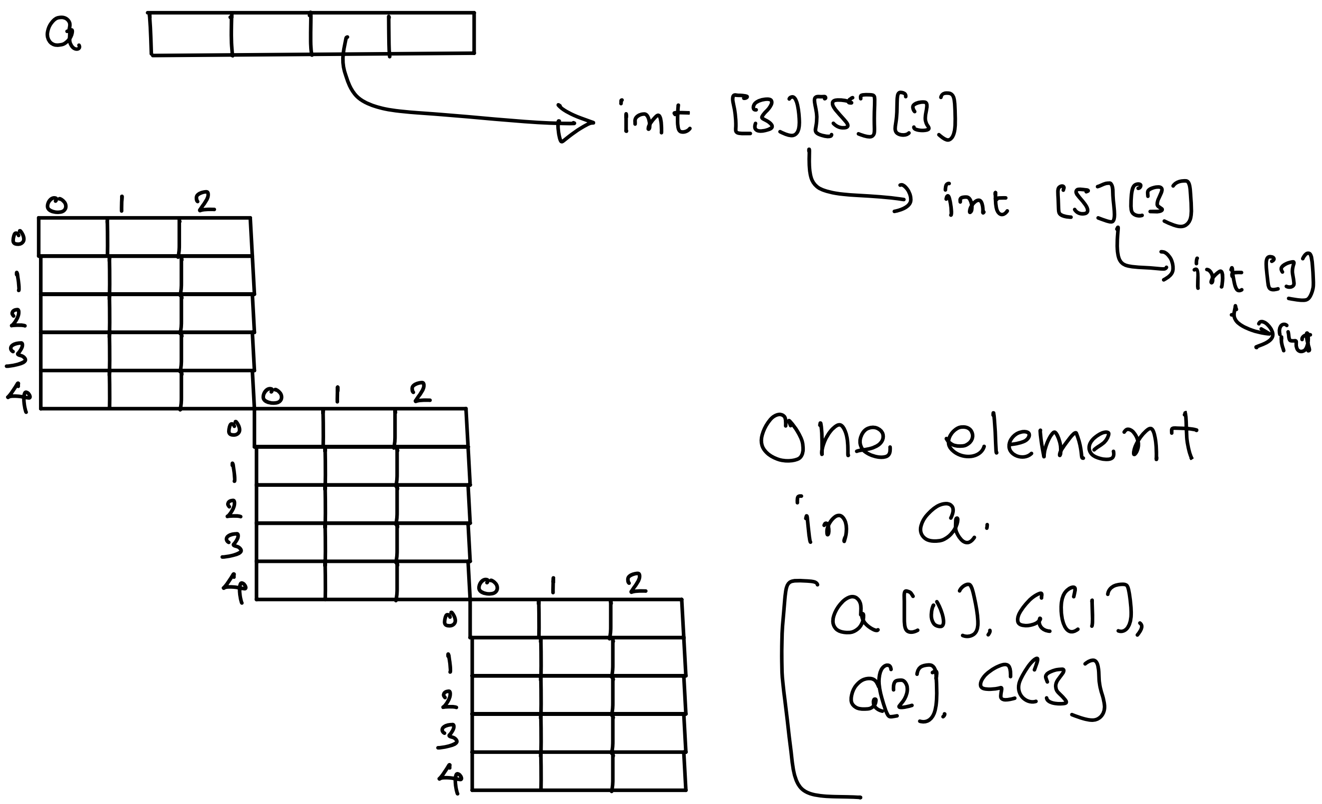
ay



4 Dimensional Array

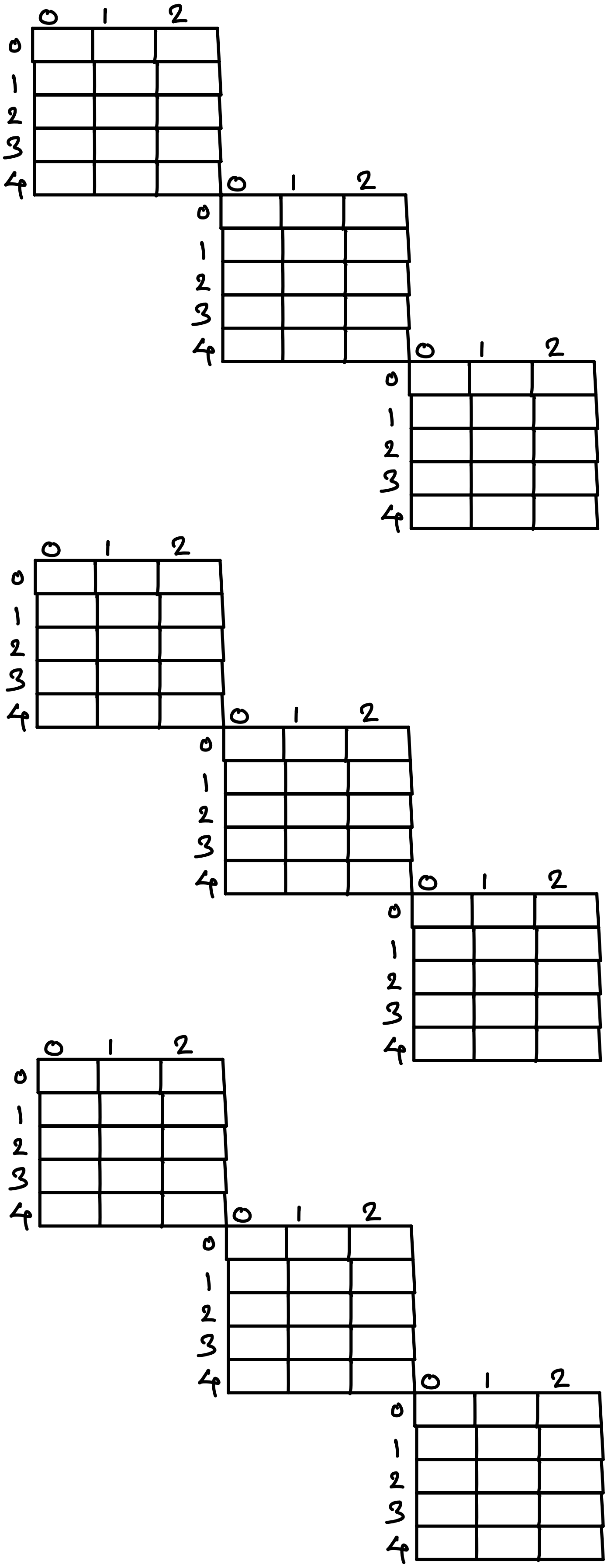
int a[4][3][5][3];

a is array 4 of array 3 of array 5 of array 3 of int.



One element in a.

a[0], a[1],
a[2], a[3]



a[0]

int [3][5][3]

---

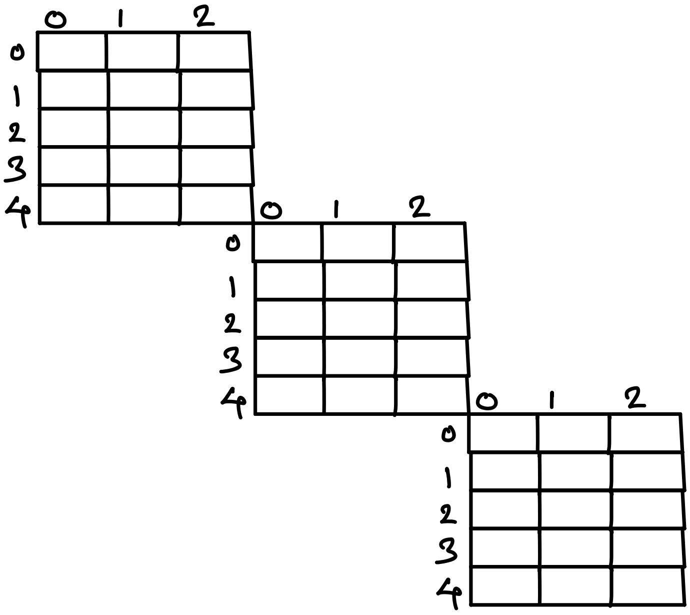
a[1]

int [3][5][3]

---

a[2]

int [3][5][3]



a[3]

int [3][5][3]

---

int $a[4]\underline{[3][5][3]}$

base addr $(a[i_1])$ = Base Addr (a) +
i_1 * sizeof(int) * 3 * 5 * 3

Base addr $(a[i_1][i_2])$ =
Base addr(a) + i_1 * sizeof(int) * 3*5*3 +
i_2 * sizeof(int) * 5 * 3 +

Base addr $(a[i_1)(i_2)(i_3])$ =
Base Addr(a) + i_1 * sizeof(int) * 3*5*3 +
i_2 * sizeof(int) * 5*3 + i_3 * sizeof(int) * 3

Base addr $(a[i_1][i_2][i_3][i_4]) =$

Base Addr$(a) + i_1 *$ sizeof(int) $* 3 * 5 * 3 +$
$i_2 *$ sizeof(int) $* 5 * 3 + i_3 *$ sizeof(int) $* 3$
$+ i_4 *$ sizeof(int)

Assume, a is k-dimensional array of int.

r^{th} dimension is N_r where $1 \leq r \leq k$

[1 D, N_1, 2 D, N_2, 3D N_3, — k-D, N_k]

int a[N_1][N_2][N_3] - - - [N_k];

let i_1, i_2, i_3 - - , i_k be k index variables.

$0 \leq i_1 < N_1$, $0 \leq i_2 < N_2$, $0 \leq i_3 < N_3$, ..., $0 \leq i_k < N_k$.

Base Addr $(a[i_1][i_2][i_3] \text{ -- } [i_k])$

$= $ Base Addr (a) $+$

$i_1 * S_{int} * N_2 * N_3 * - - * N_k +$

$i_2 * S_{int} * N_3 * N_4 * \cdots * N_k +$

$i_3 * S_{int} * N_4 * N_5 * - - * N_k +$

$\vdots$

$i_k * S_{int}$.

$= \text{Base Addr}(a) +$

$$\left[\sum_{r=1}^{r=k-1} i_r \times \left(\prod_{s=r+1}^{s=k} N_s\right)\right] * S_{int} +$$

$i_k * S_{int}$

↪ Important for freshers.

↪ C + DSA interview.