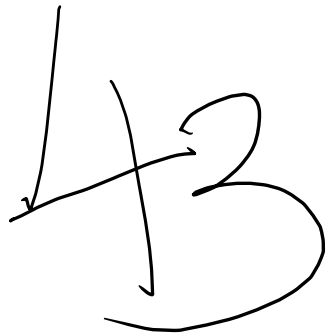
Let’s solve it



For one dimensional array arr1 is of datatype int

int arr1d[10];

arr1d[i] ⇒ *(arr1d+i)

arr1d[3]
*(arr1d+3)

arr1d is a pointer to single integer.

arr1d+1 ⇒ next element (just one)

arr1d+2 ⇒ You skip 2 elements of type int

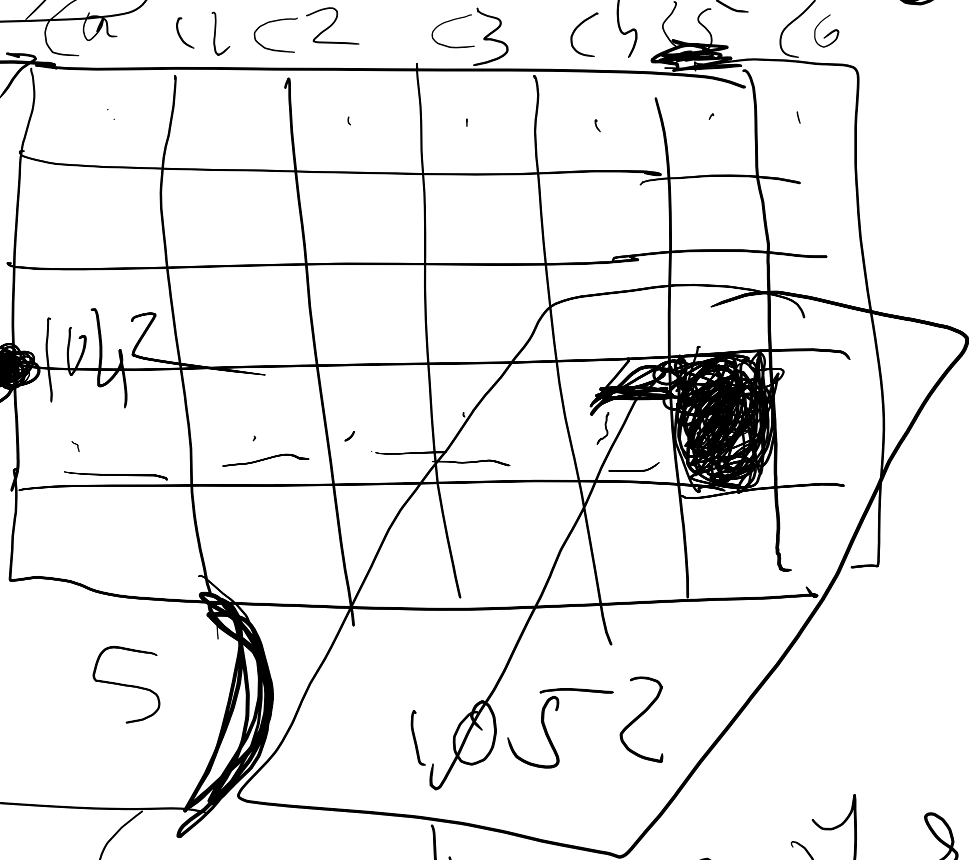
From many (i.e 10 seperate elements) ⇒ stepdown one less one single seperate element

For two dimensional array

int arr[5][7]

actually has datatype int (*7) no of colums
a row

element arr[3][5] 1000 row 0
of rows 3 (4th)
columns (6th)



pointer arithmetic

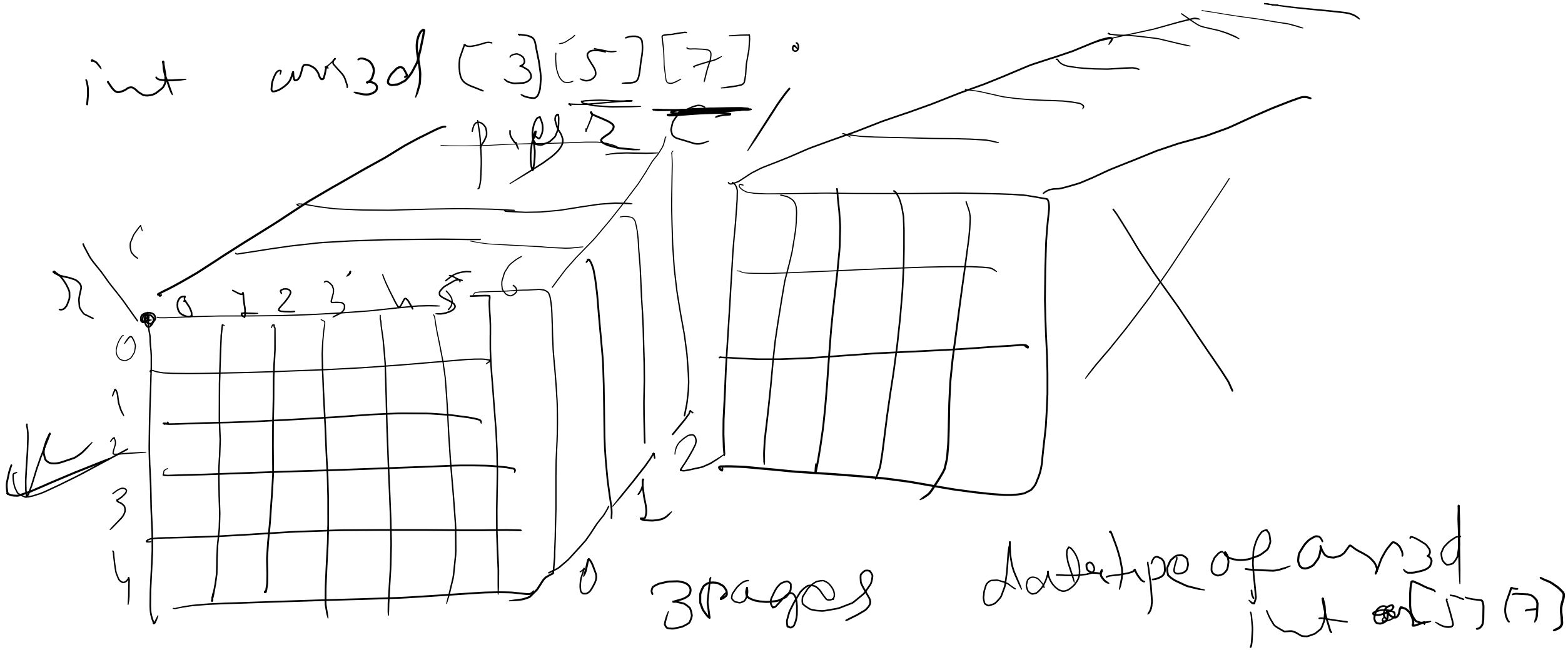
((arr + 3) + 5)

1052

From 3 row x 7 colums => (7 columns) 1 row

down
Grading

arr + 3 to jum 3 rows


int arr3d[3][5][7];
pages
0 1 2 3 4 5 6
0
1
2
3
4
0
1
2
3 pages
datatype of arr3d
int arr[5][7]
arr3d + 1
&arr3d[0][0][0]

* has multiple purpose — asterisc

(1) declaring pointer
int *ptr;

(2) ptr = &x;
printf("%d", *ptr);
accessing data of x via pointer ptr (dereferencing)

(3) downgrade (going a step down in multidimension)
datatype *(arr2d + i)
datatype of row
datatype of single element

2 dim

Base addr → 1000 Row level

A — [arr2d + 3] + 5 → rows

1042

Vs

comp maths

B — *(arr2d + 3) + 5

1042

1000

Regular Maths

1042
+ 5 × (2)

1042
+ 10 bytes

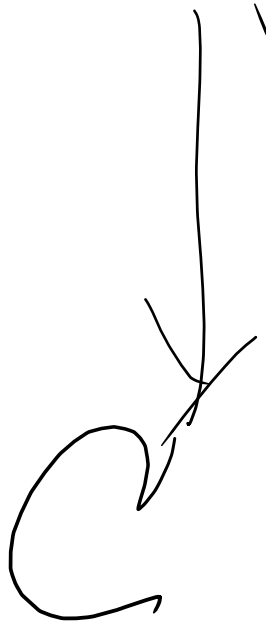
1052

~~elements~~

1000 + 3 rows ⇒ 1000 + 3 (7 elements)

⇒ 1000 + 3 (7 × 2 bytes for int)

Row major Vs column major



all the elements of a row will be in sequence inside memory, - - -

Comp. Maths

printf("%d" arr[3][5])

$*(*(arr2d+3)+5)$

int arr[5][7];

Wandfore

downgrading?

Regular maths

arr2d + (3 * 7) * 2 + (5 * 2)

no. of columns

capacity of a single row

one element of 2d arr

  1000
+   42
+   10

  1052