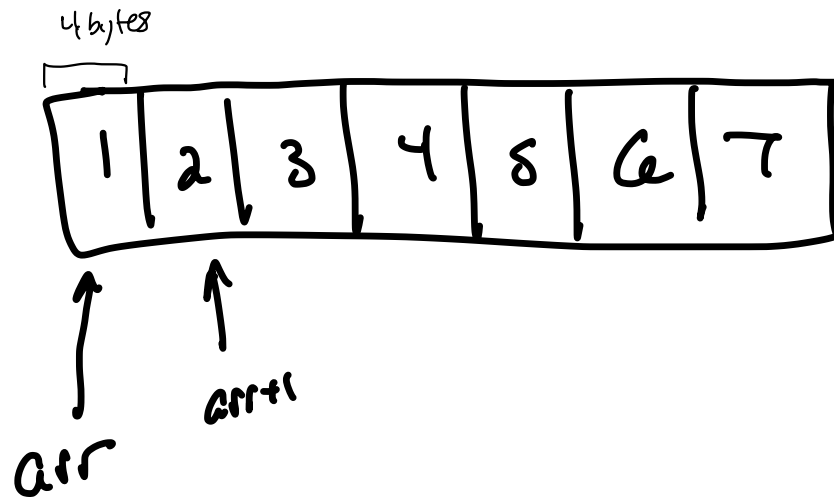


C - MULTIDIMENSIONAL ARRAYS

2D ARRAYS

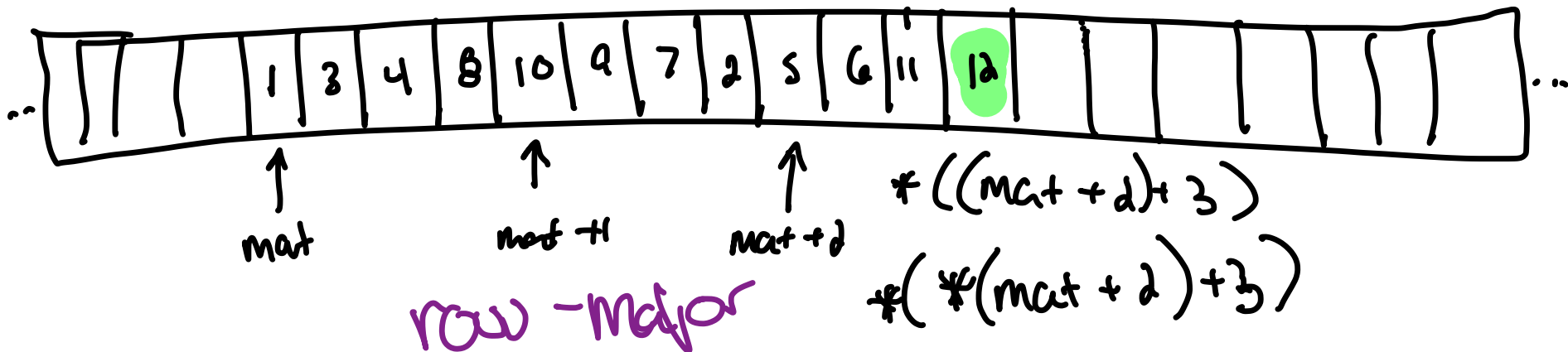
- Example: `int mat[2][3];`
 - 2 rows, 3 columns
- Can initialize when declaring:
 - `int mat[2][3] = {{1, 3, 4}, {8, 2, 5}};`
- Access elements: `mat[row_num][colnum]`
- Declared on the stack
- One contiguous block of memory underneath



1	3	4	8
10	9	7	2
5	6	11	12

how we think of it

how it's actually stored



2D ARRAYS (CONT.)

- $\text{mat}[i][j]$ = an element of the 2d array
- mat = address of first element
- $\text{mat}[i] = \&\text{mat}[i]$ = address to first element of row i
- $\text{mat}+i = *(\text{mat}+i) = \wedge$ address to first element of row i
- Think of $\text{mat}[i]$ as a pointer to an array (row)
- Doing pointer arithmetic on mat advances by the whole length of the subarrays

2D ARRAYS (CONT.)

- Can also use pointer arithmetic to access into certain col
- $\ast(\text{mat}[i] + j) = \text{mat}[i][j]$
- $\ast(\ast(\text{mat} + i) + j) = \text{mat}[i][j]$
 - recall $\ast(\text{mat} + i)$ is memory address to start of row i
 - advance this by j cols, then dereference to get actual value stored
- What about $\text{mat} + i + j$?