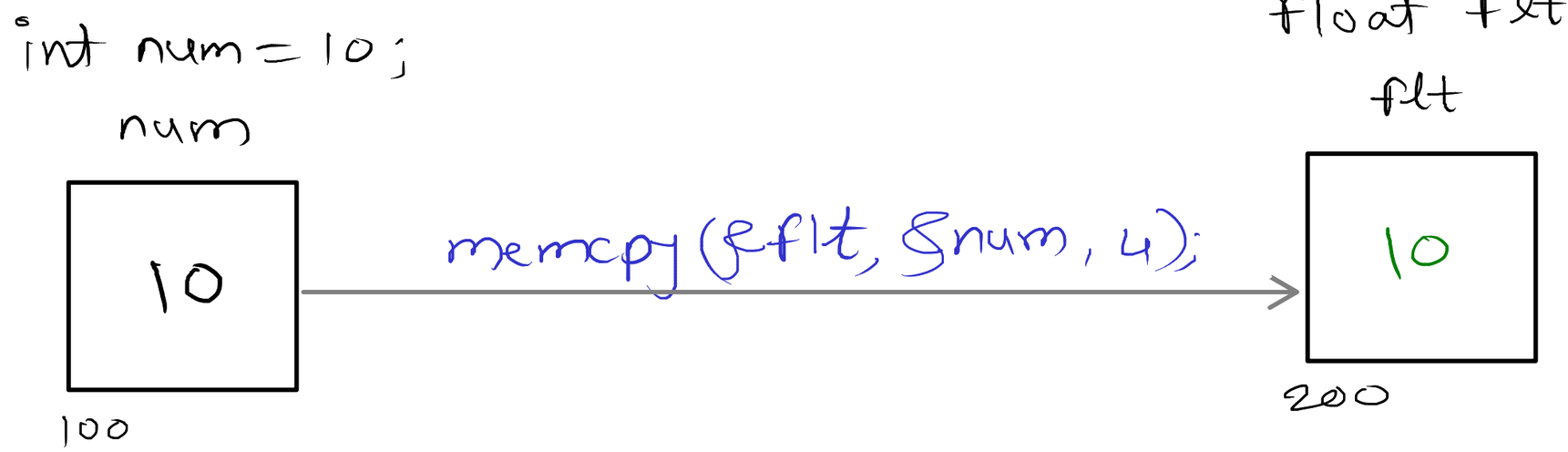
int num = 10;

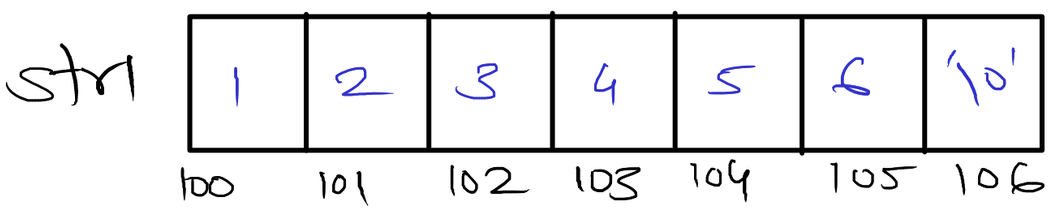


&flt = 200
↳ float *

(int *)&flt
↳ int *

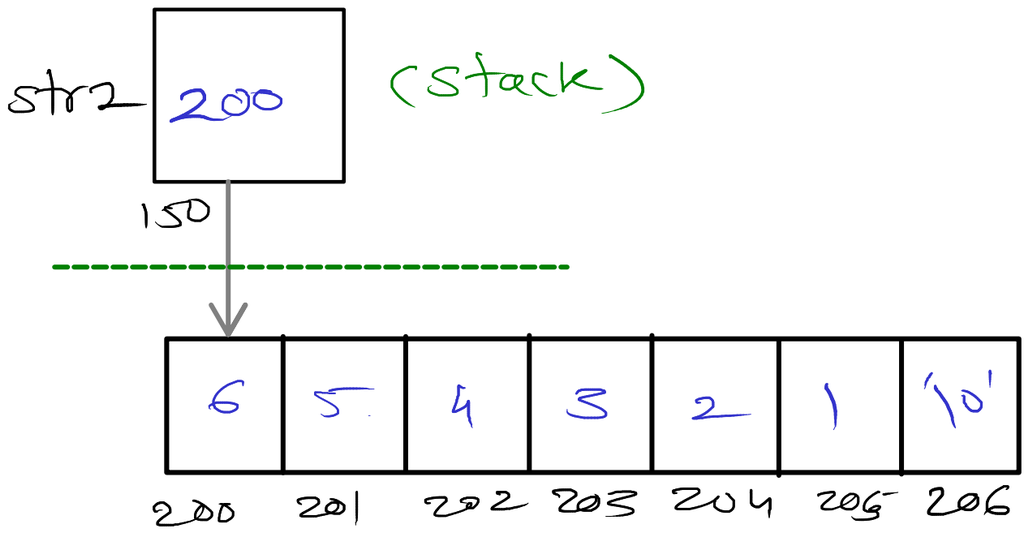
*(int *)&flt
↳ int

char str1[] = "123456";



(stack)

char * str2 = "654321";



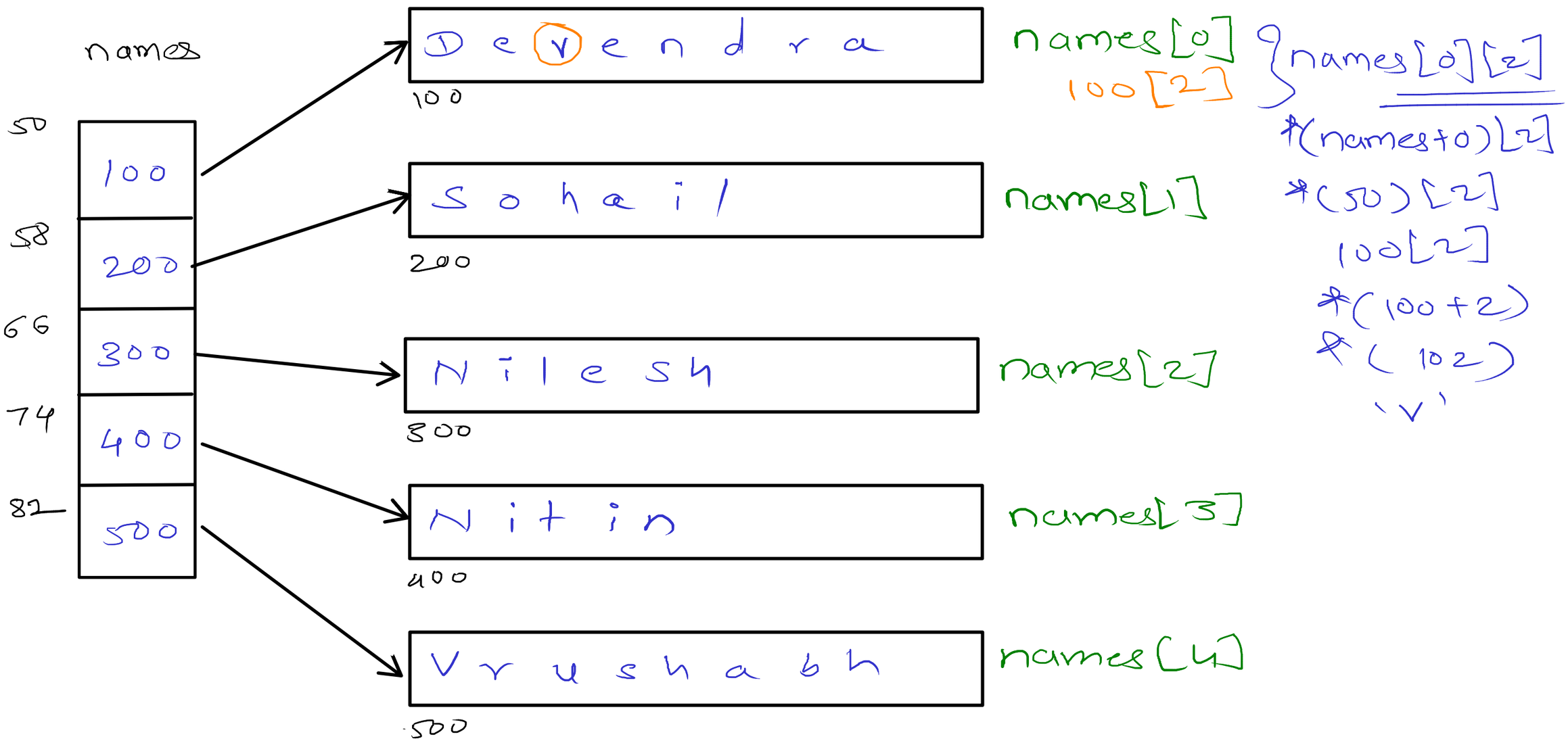
(no data)

memcpy(str2, str1, sizeof(str1)); - error

memcpy(str1, str2, sizeof(str1));

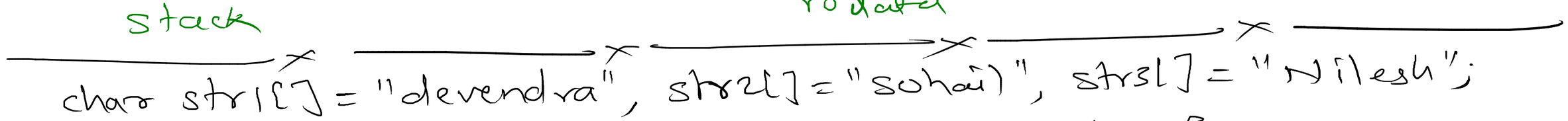
char *names[] = {

↳ Array of character pointers



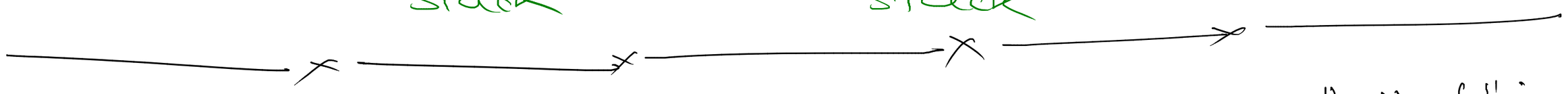
```
char *names[] = { "devendra", "sohail", "Nilesh", "Nitn", "Vrushabh" };
```

stack ro data



```
char str1[] = "devendra", str2[] = "sohail", str3[] = "Nilesh";
char *names[] = { str1, str2, str3 };
```

stack stack



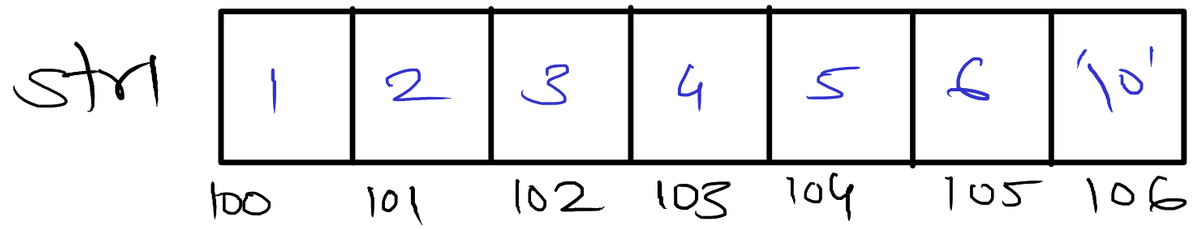
```
char *str1 = "devendra", *str2 = "sohail", *str3 = "Nilesh";
char *names[] = { str1, str2, str3 };
```

names → stack

str1, str2, str3 → steek

"devendra", "sohail", "Nilesh" → ro. data

```
char str1[] = "123456";
```



(stack)

str1 – base address

(address of first element of array)

SF = sizeof(str1[0])

str1+1 = 100 + SF
= 100 + 1
= 101

char *ptr = str1
↳ character pointer

&str1 – address of array

(address of whole array)

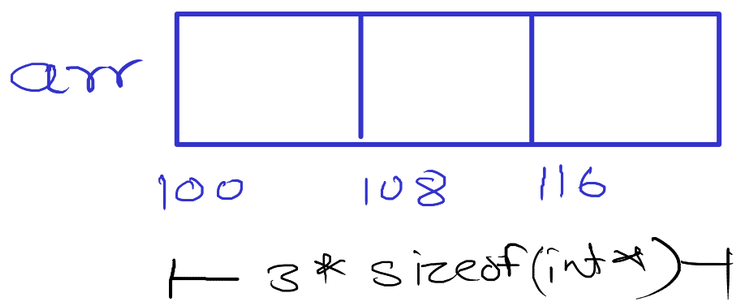
SF = sizeof(str1)

&str1+1 = 100 + SF
= 100 + 7
= 107

char (*ptr)[7] = &str1
↳ array pointer

`int *arr[3]`

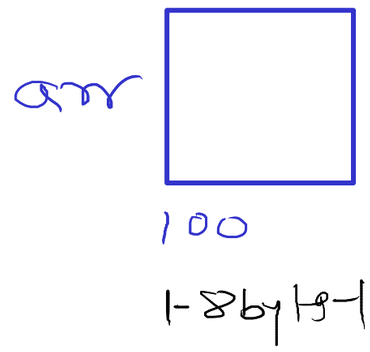
- Array of interger pointers



- we can store 3 addresses of type 'integer pointers

`int (*arr)[3]`

- Array pointer



- we can store single address of type array pointer.

`int arr[5]` ⟶ 1D Array

↳ no of elements in array

`int arr[3][4]` ⟶ 2D Array (matrix)

↳ no. of columns

↳ no of rows

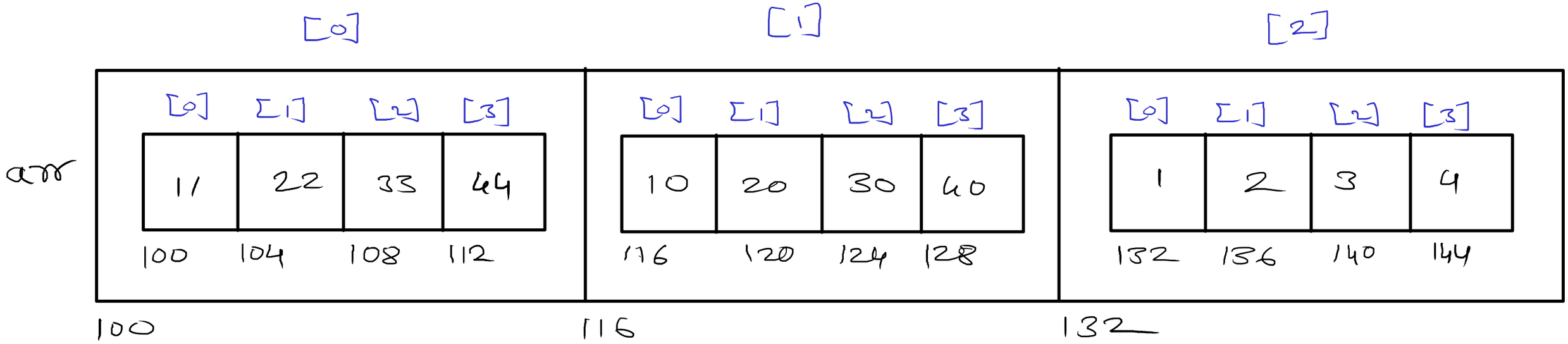
	0	1	2	3
0	11 [0][0]	22 [0][1]	33 [0][2]	44 [0][3]
1	10 [1][0]	20 [1][1]	30 [1][2]	40 [1][3]
2	1 [2][0]	2 [2][1]	3 [2][2]	4 [2][3]

arr

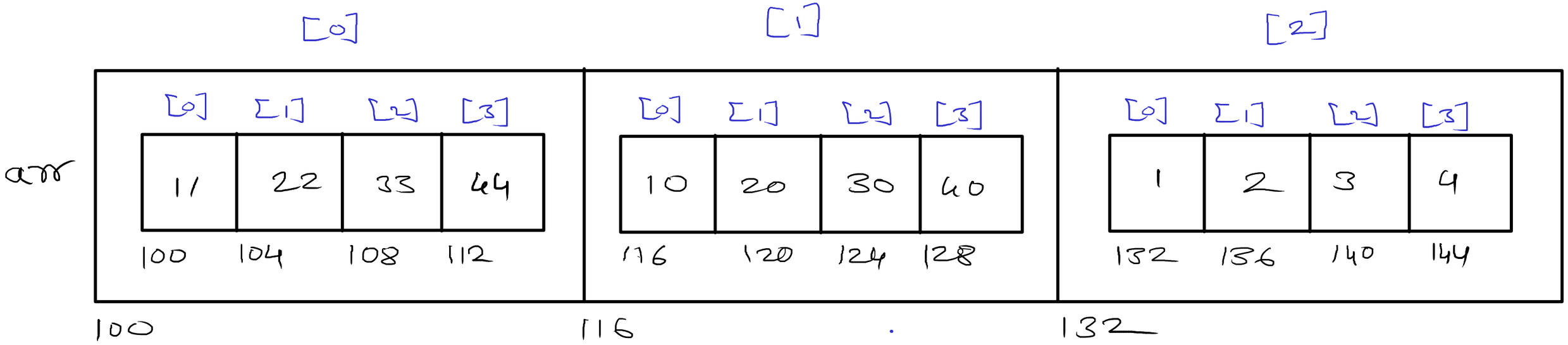
`arr[i][j]`

where i = 0, 1, 2

j = 0, 1, 2, 3



`int size = 5;`
`int arr[size];` { `int (*arr)[col]` —{.



1D Array → arr[i] = *(arr+i) — i^{th} element

2D Array → arr[i] = *(arr+i) — i^{th} element

arr[0] — 1st 1D array

arr[1] — 2nd 1D array

arr[2] — 3rd 1D array

arr[i][j] = *(arr+i)[j] = *(*(arr+i)+j)

arr[0][0] = 11

((arr+0)+0)

((100+0)+0)

*(*100+0)

*(100+0) = *100

= 11

arr[1][2] = 30

((arr+1)+2)

((100+1)+2)

*(*116+2)

*(116+2)

*124 = 30

```
int arr[2][3][3];
```

blocks row col

e.g. $\begin{bmatrix} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}, \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \end{bmatrix}$

arr[i][j][k]

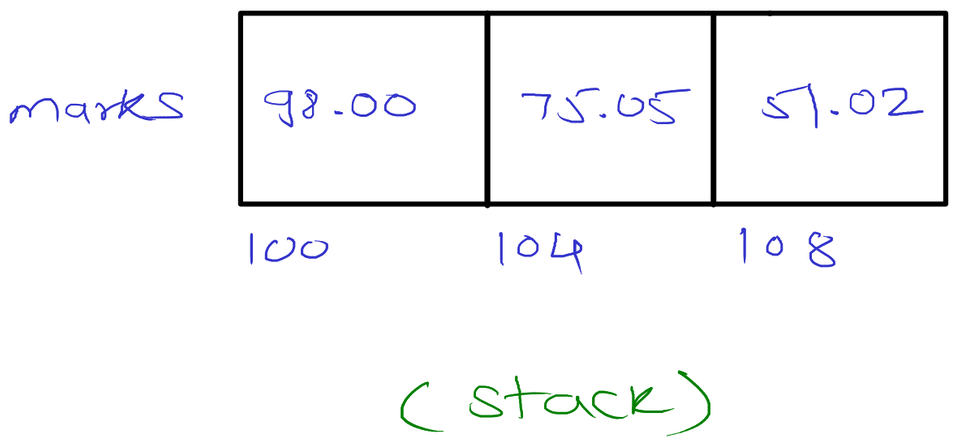
i = 0, 1

j = 0, 1, 2

k = 0, 1, 2

```
int (*ptr)[3][3] = arr;
```

```
float marks[3];
```




```
float *marks = (float *)malloc(12);
```