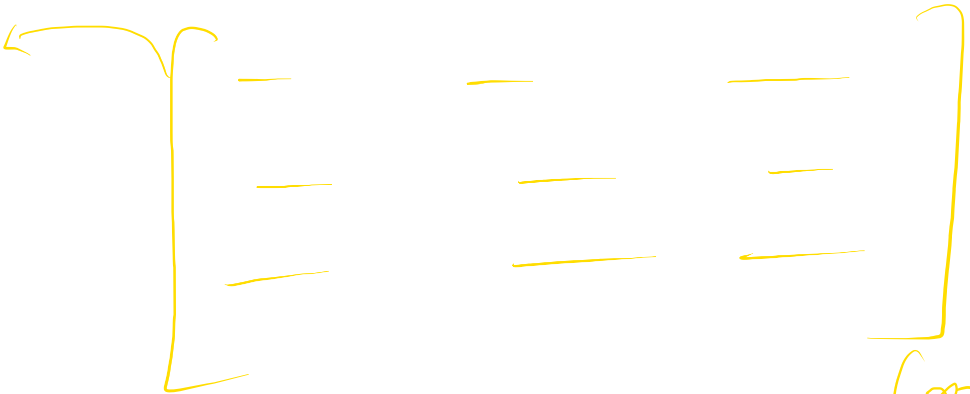
2 D Arrays

↳ 2 Dimen

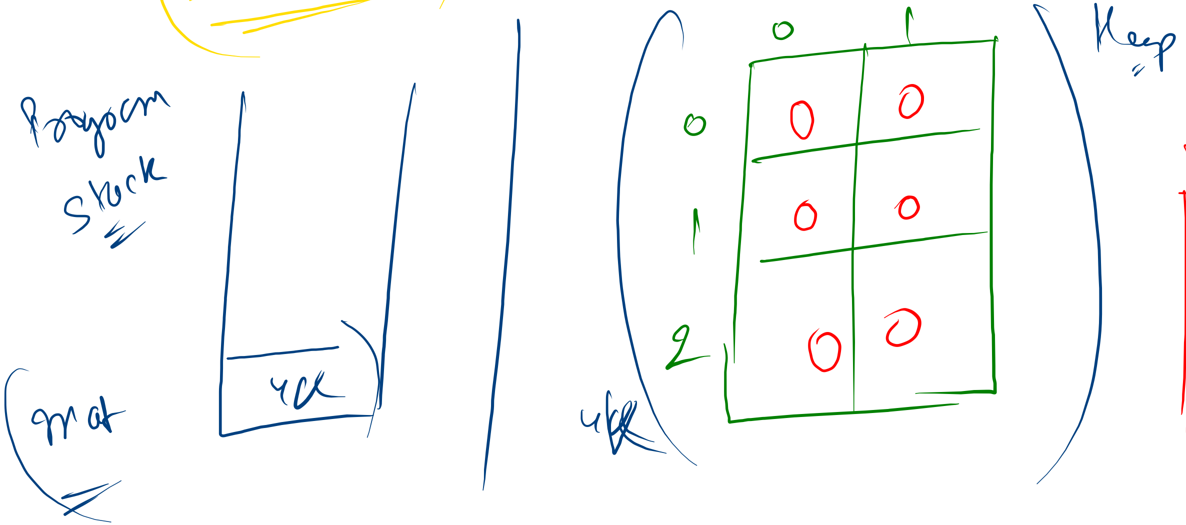


(row x col) → order

(Declaration) → int mat[][] = new int[3] [2]

n row

n col



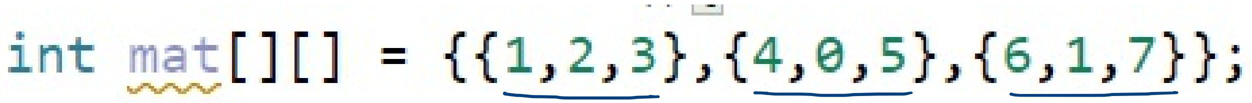
NOTE

mat. length => n.rows

mat[0]. length => ncols

→ why?

↳ memory management



```
int mat[][] = {{1,2,3},{4,0,5},{6,1,7}};
```

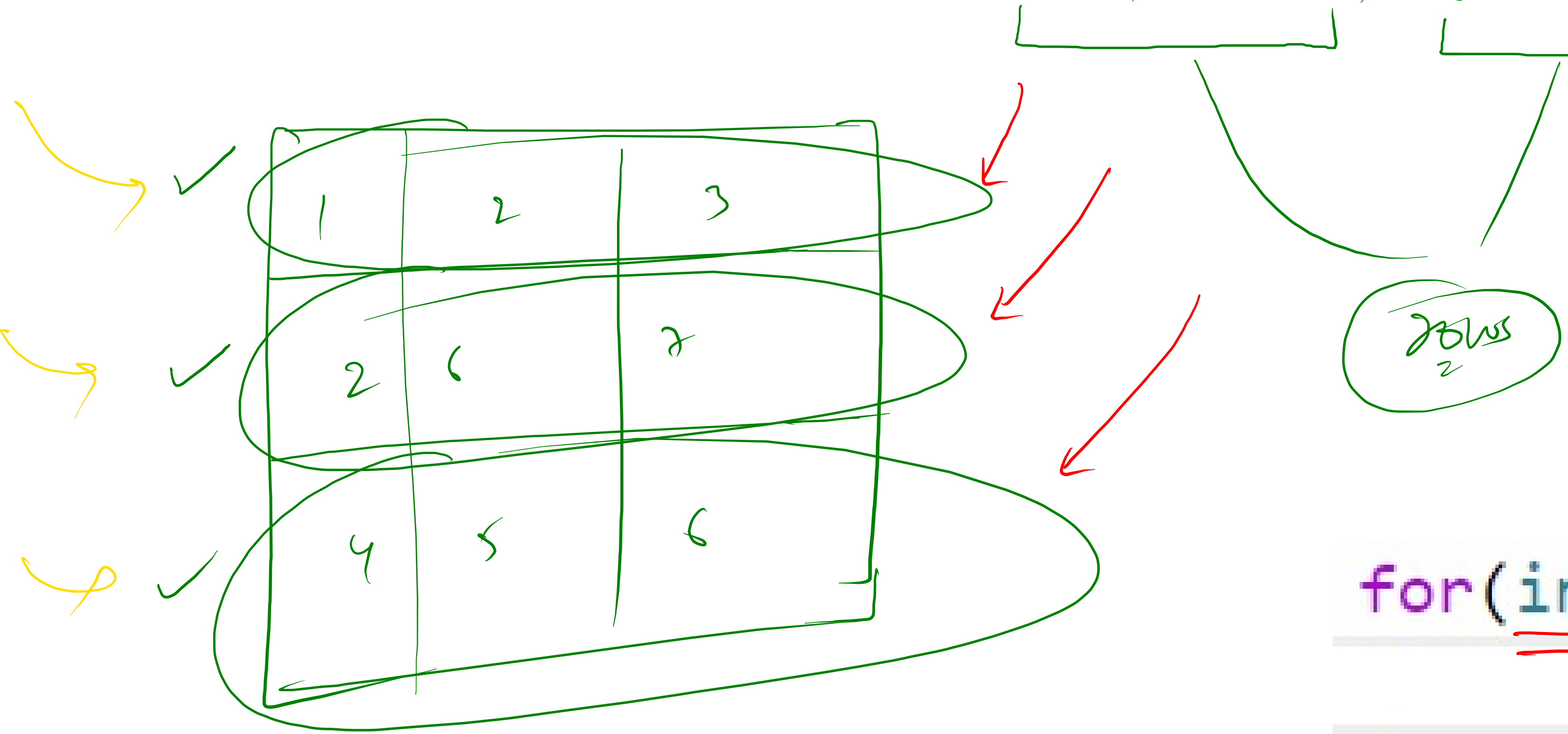
(nc-1)

rows

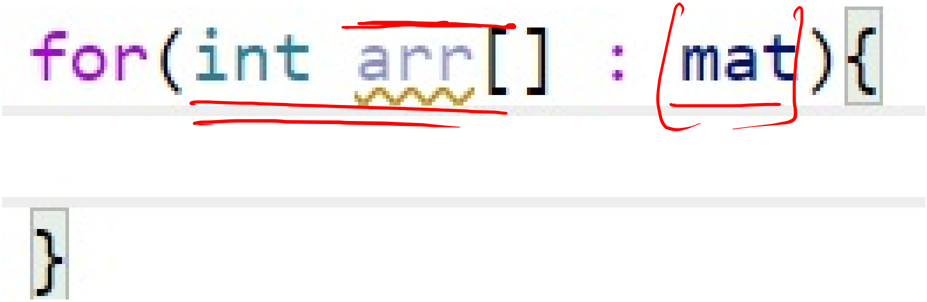
$$\begin{array}{c|ccc} & (0 & 1 & 2) \rightarrow \text{cols} \\ \hline 0 & 1 & 2 & 3 \\ 1 & 4 & 0 & 5 \\ 2 & 6 & 1 & 7 \end{array}$$

(nr-1)

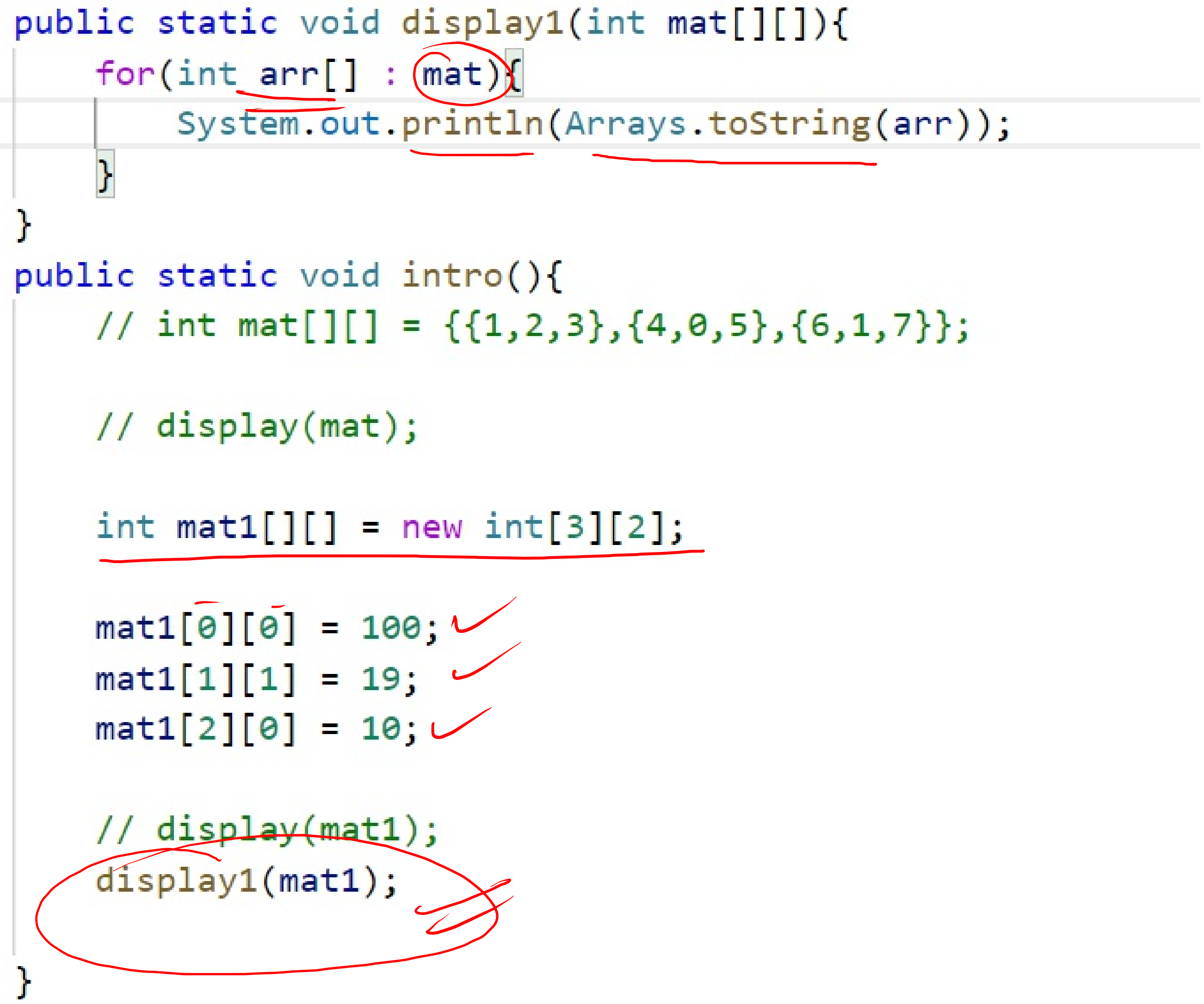
int mat[][] = { {1, 2, 3}, {2, 6, 7}, {4, 5, 6}};



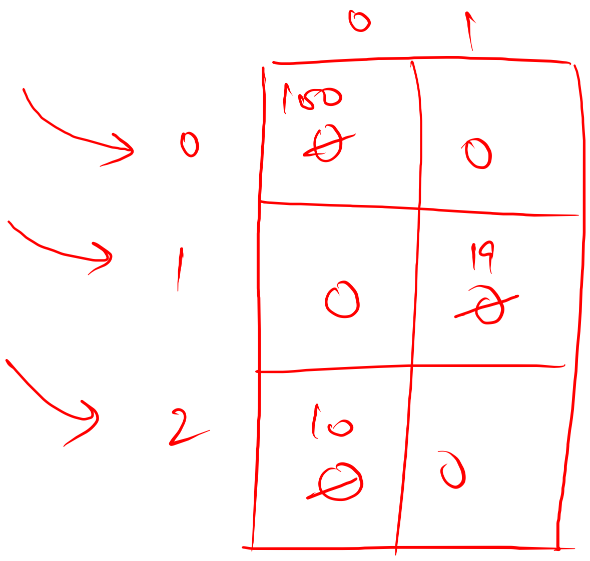
1	2	3
2	6	7
4	5	6



```
for(int arr[] : mat){

}
```



```java
public static void display1(int mat[][]){
    for(int arr[] : mat){
        System.out.println(Arrays.toString(arr));
    }
}
public static void intro(){
    // int mat[][] = {{1,2,3},{4,0,5},{6,1,7}};

    // display(mat);

    int mat1[][] = new int[3][2];

    mat1[0][0] = 100;
    mat1[1][1] = 19;
    mat1[2][0] = 10;

    // display(mat1);
    display1(mat1);
}
```



	0	1
0	100 ~~0~~	0
1	0	19 ~~0~~
2	10 ~~0~~	0

[100 , 0]

[0 , 19]

[10 , 0]

	0	1	2	3
0	11	12	13	14
1	21	22	23	24

① Input

② Display

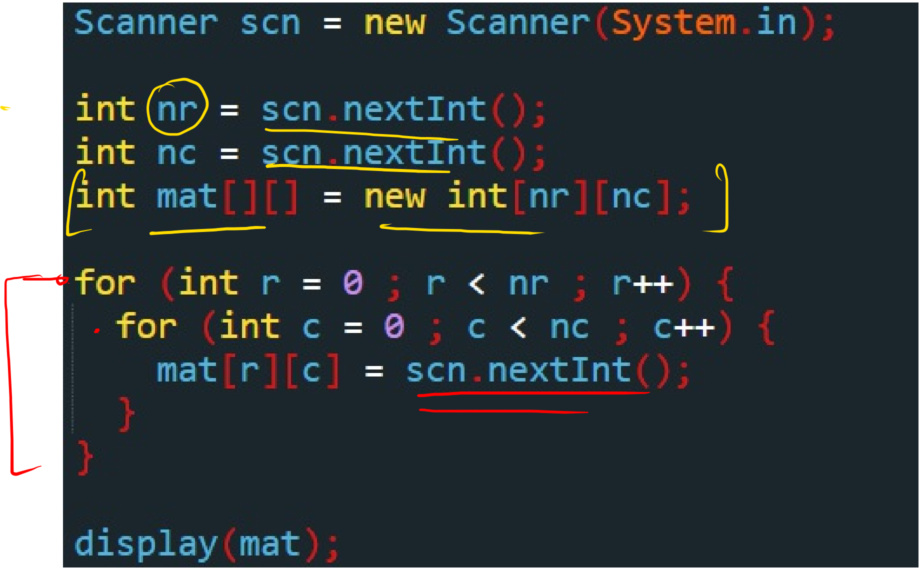
$$\begin{bmatrix} 11 & 12 & 13 & 14 \\ 21 & 22 & 23 & 24 \end{bmatrix}$$

nr = 2

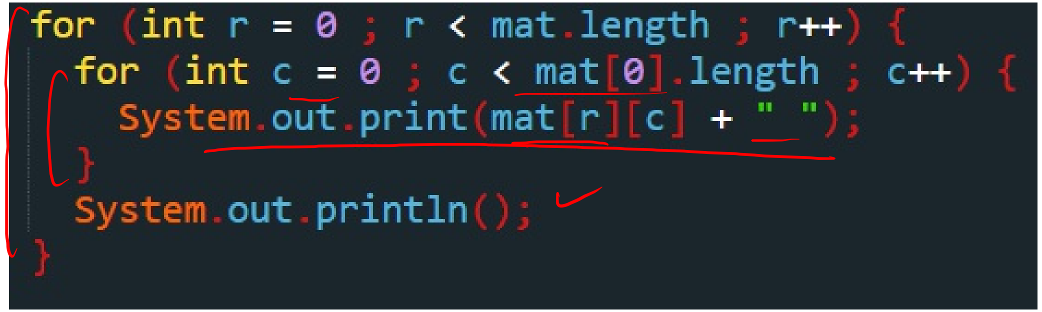
nc = 4

$$\begin{array}{c|cccc} & 0 & 1 & 2 & 3 \\ \hline 0 & 11 & 12 & 13 & 14 \\ 1 & 21 & 22 & 23 & 24 \end{array}$$

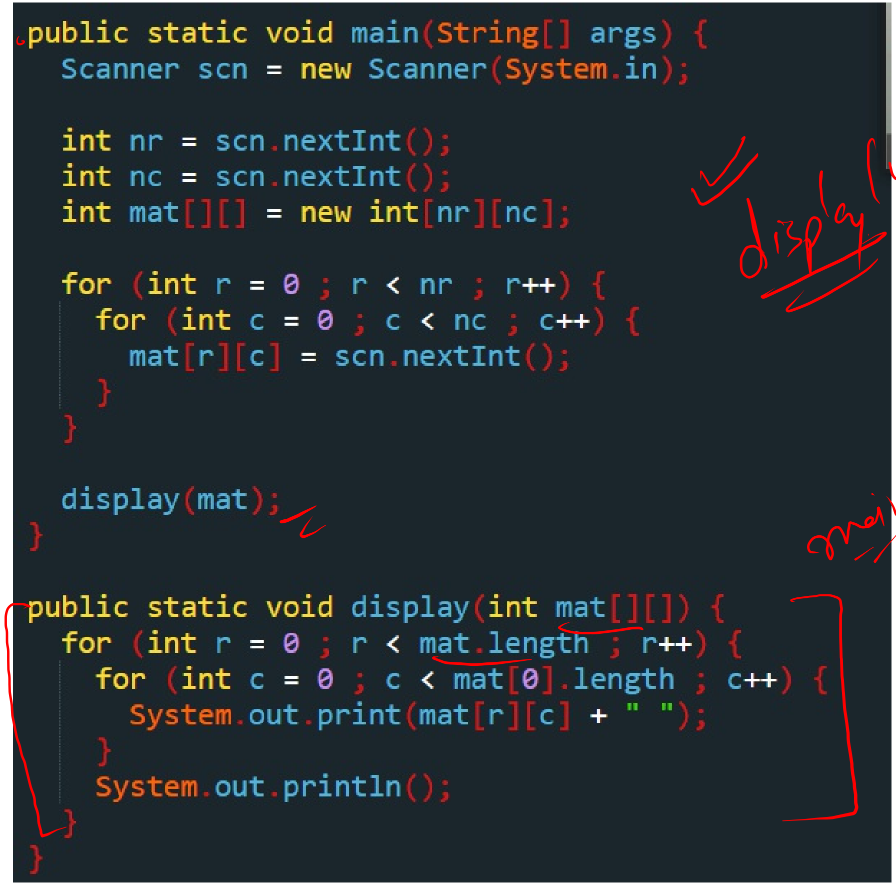
$$\begin{bmatrix} 11 & 12 & 13 & 14 \\ 21 & 22 & 23 & 24 \end{bmatrix}$$



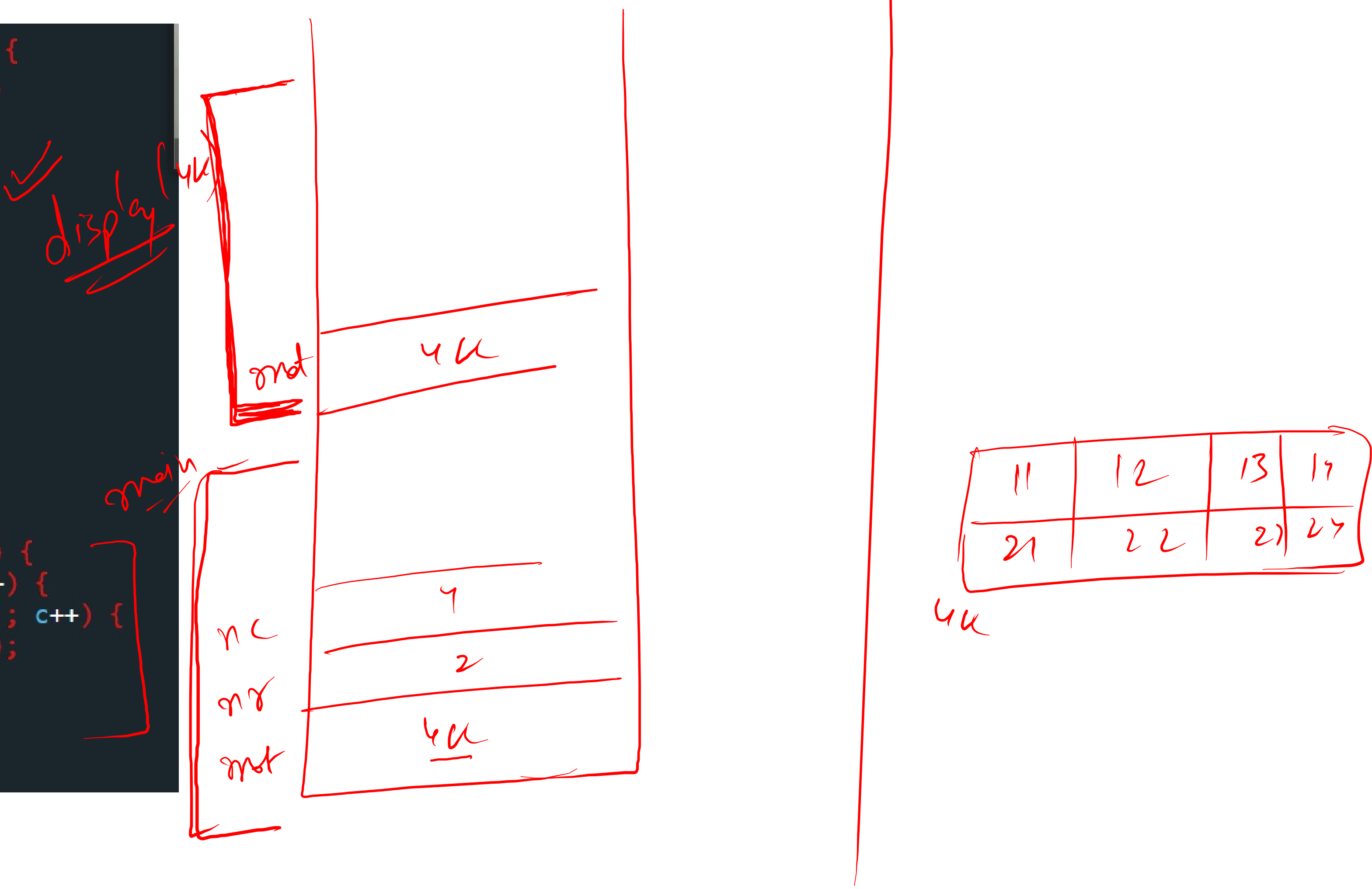
```java
Scanner scn = new Scanner(System.in);

int nr = scn.nextInt();
int nc = scn.nextInt();
int mat[][] = new int[nr][nc];

for (int r = 0 ; r < nr ; r++) {
   for (int c = 0 ; c < nc ; c++) {
      mat[r][c] = scn.nextInt();
   }
}

display(mat);
```



```java
for (int r = 0 ; r < mat.length ; r++) {
   for (int c = 0 ; c < mat[0].length ; c++) {
      System.out.print(mat[r][c] + " ");
   }
   System.out.println();
}
```



```java
public static void main(String[] args) {
  Scanner scn = new Scanner(System.in);

  int nr = scn.nextInt();
  int nc = scn.nextInt();
  int mat[][] = new int[nr][nc];

  for (int r = 0 ; r < nr ; r++) {
    for (int c = 0 ; c < nc ; c++) {
      mat[r][c] = scn.nextInt();
    }
  }

  display(mat);
}

public static void display(int mat[][]) {
  for (int r = 0 ; r < mat.length ; r++) {
    for (int c = 0 ; c < mat[0].length ; c++) {
      System.out.print(mat[r][c] + " ");
    }
    System.out.println();
  }
}
```

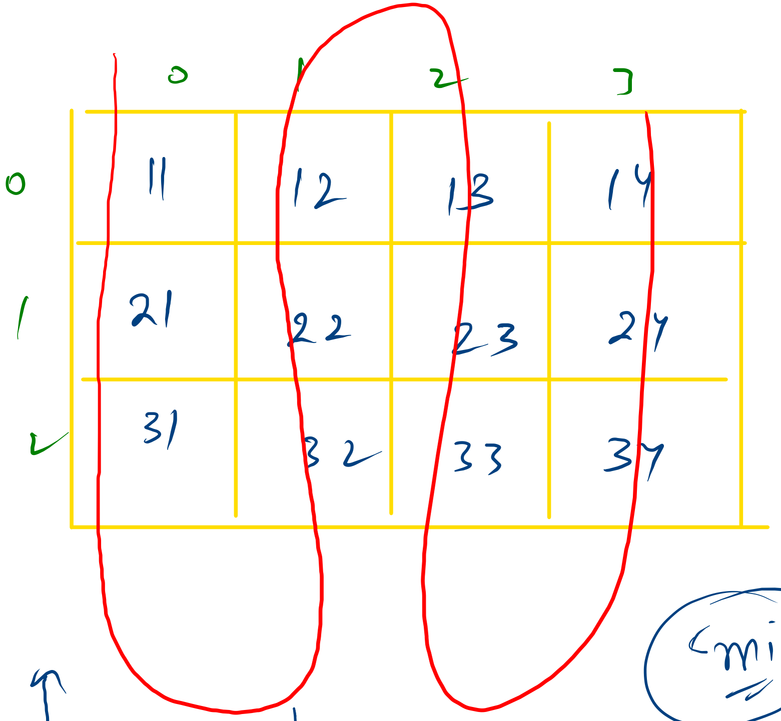


11	12	13	14
21	22	23	24

Traversal

→ nr

nc —



$$\begin{array}{c|cccc} & 0 & 1 & 2 & 3 \\ \hline 0 & 11 & 12 & 13 & 14 \\ 1 & 21 & 22 & 23 & 24 \\ 2 & 31 & 32 & 33 & 34 \end{array}$$

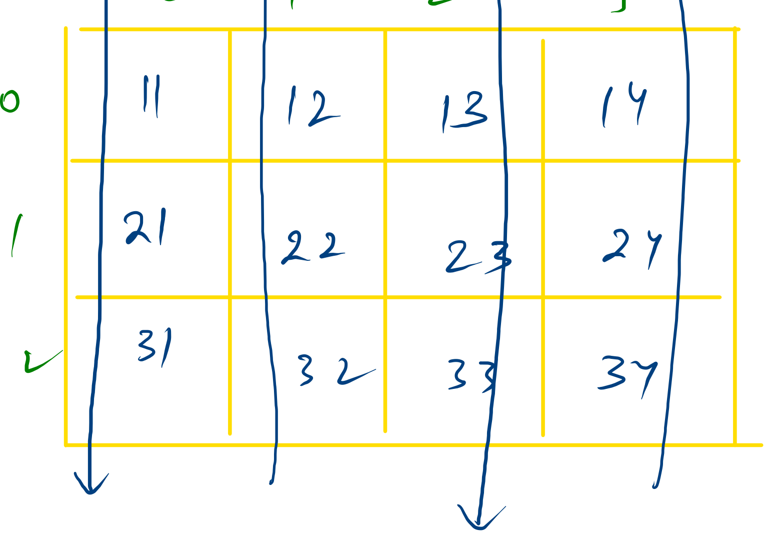
11
21
31
32
22
12
13
23
33
34
24
14

<min

Approach

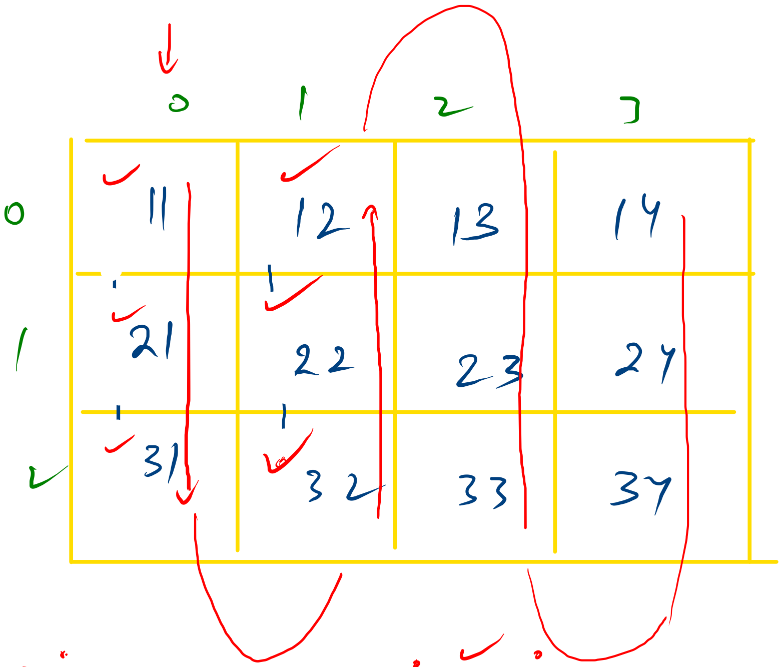
Col => even
→ top → bottom

col => odd
→ bottom → top



$$\begin{array}{c|cccc} & 0 & 1 & 2 & 3 \\ \hline 0 & 11 & 12 & 13 & 14 \\ 1 & 21 & 22 & 23 & 24 \\ 2 & 31 & 32 & 33 & 34 \end{array}$$

nr = 3



	0	1	2	3
0	11	12	13	14
1	21	22	23	24
2	31	32	33	34

Col → 0 row = 0, 1, 2

Col → 1 row = 2, 1, 0

top → bottom

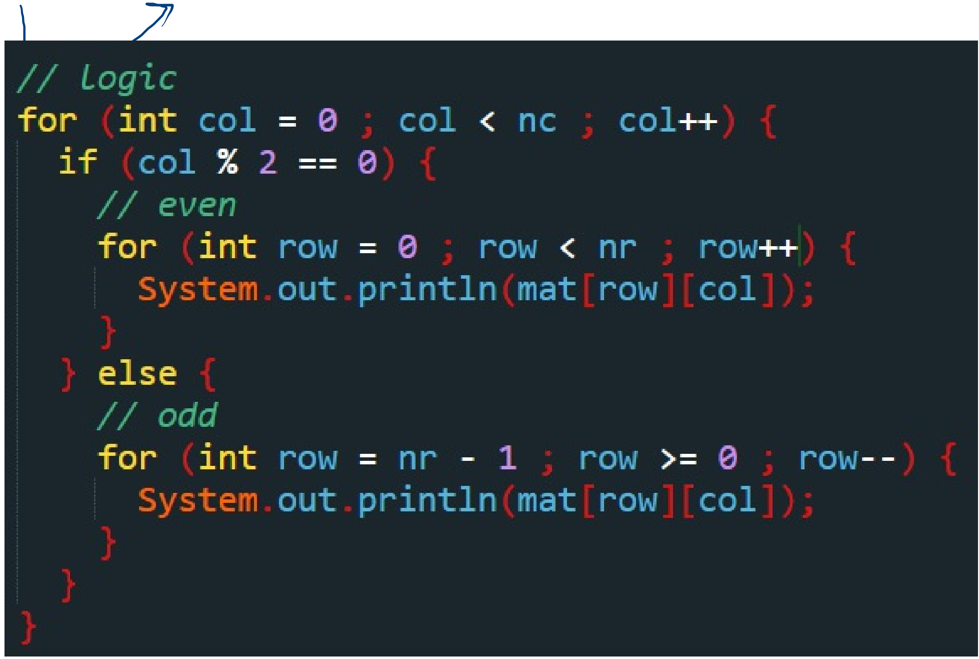
row → 0 → len-1

++

bottom → top

row → len-1 → 0

11
21
31
32
22
12



```
// logic
for (int col = 0 ; col < nc ; col++) {
  if (col % 2 == 0) {
    // even
    for (int row = 0 ; row < nr ; row++) {
      System.out.println(mat[row][col]);
    }
  } else {
    // odd
    for (int row = nr - 1 ; row >= 0 ; row--) {
      System.out.println(mat[row][col]);
    }
  }
}
```

$res[1][3] = (0 \cdot 0) + (20 \cdot 2) + (0 \cdot 0) \Rightarrow 40$

$$mat1 = \begin{array}{c|ccc} & 0 & 1 & 2 \\ \hline 0 & 10 & 0 & 0 \\ \rightarrow 1 & 0 & 20 & 0 \end{array}$$

$$mat_2 = \begin{array}{c|cccc} & 0 & 1 & 2 & 3 \\ \hline 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 2 \\ 2 & 1 & 1 & 0 & 0 \end{array}$$

$(nr_1 \times nc_1)$

(2×3)

$(nr_2 \times nc_2)$

(3×4)

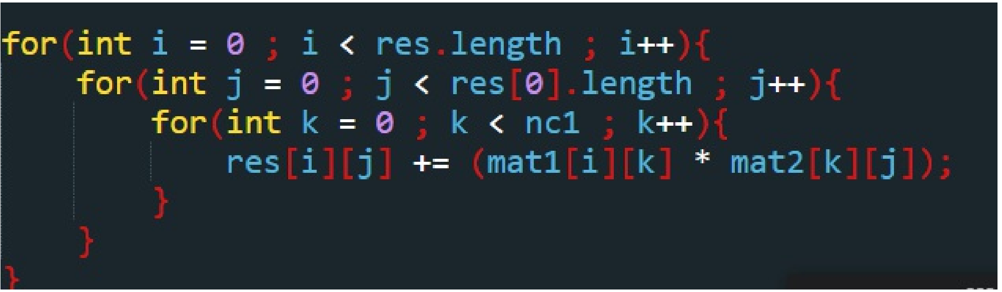
$i = 1$

$j = 3$

$k = 0, 1, 2$

$$\begin{array}{c|cccc} i \backslash j & 0 & 1 & 2 & 3 \\ \hline 0 & & & & \\ 1 & & & 20 & 40 \end{array}$$

2×4

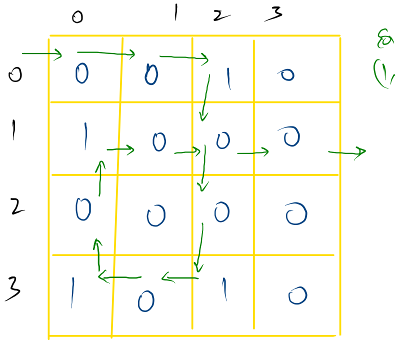


```
for(int i = 0 ; i < res.length ; i++){
    for(int j = 0 ; j < res[0].length ; j++){
        for(int k = 0 ; k < nc1 ; k++){
            res[i][j] += (mat1[i][k] * mat2[k][j]);
        }
    }
}
```

O => nr

O => nc

Exit point?



	0	1	2	3
0	0	0	1	0
1	1	0	0	0
2	0	0	0	0
3	1	0	1	0

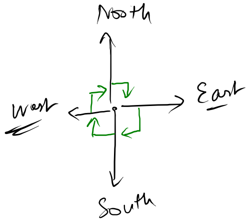
&

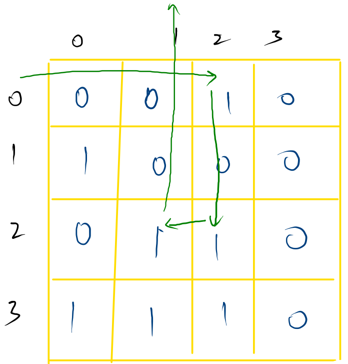
(1,3)

1 → wall → (90° right)

Entry => (0,0)

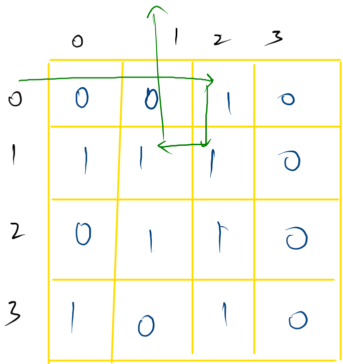
Dir => East



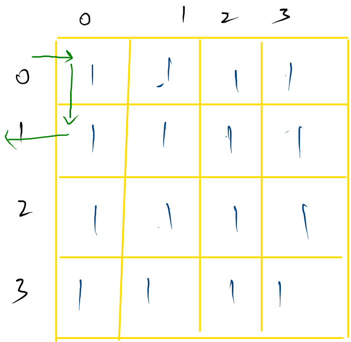


	0	1	2	3
0	0	0	1	0
1	1	0	0	0
2	0	1	1	0
3	1	1	1	0

(0,1)

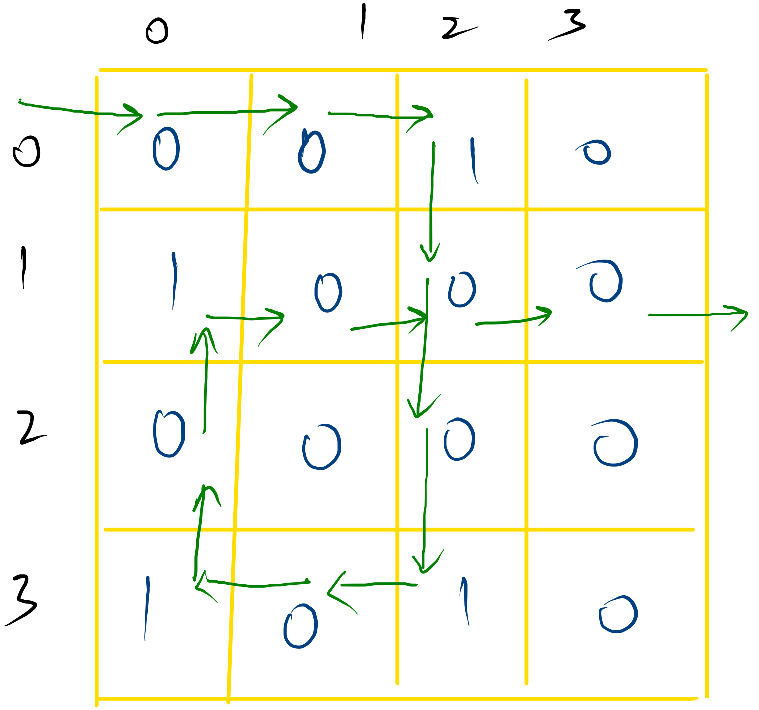


	0	1	2	3
0	0	0	1	0
1	1	1	1	0
2	0	1	1	0
3	1	0	1	0



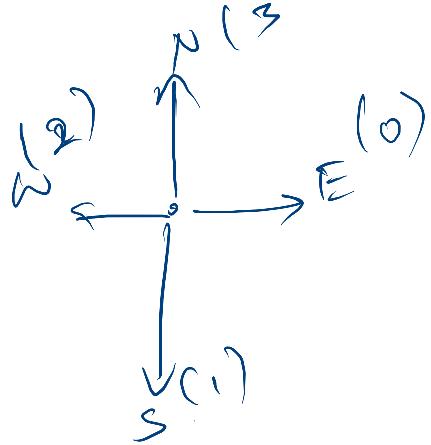
	0	1	2	3
0	1	1	1	1
1	1	1	1	1
2	1	1	1	1
3	1	1	1	1

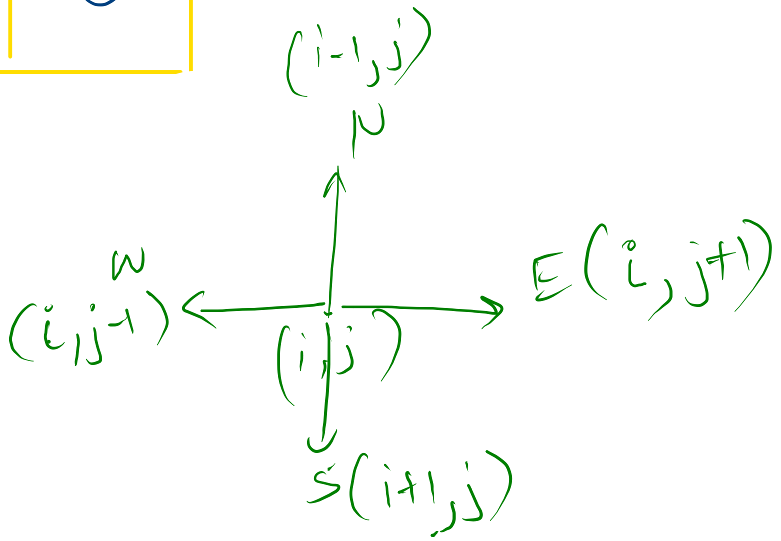
(1,0)



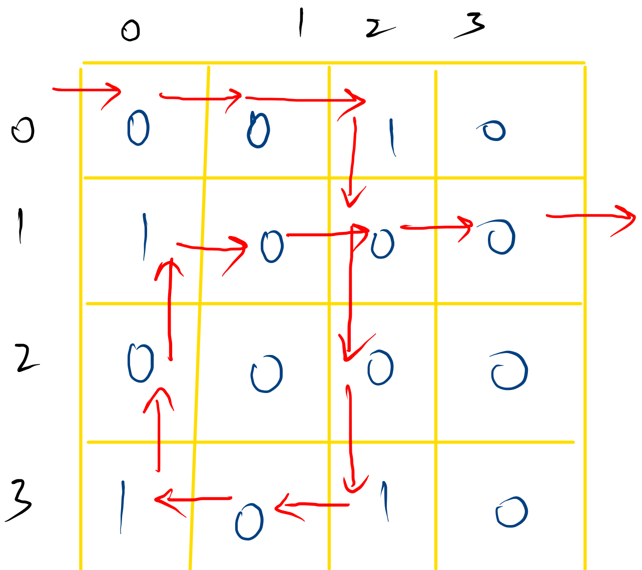
	0	1	2	3
0	0	0	1	0
1	1	0	0	0
2	0	0	0	0
3	1	0	1	0

Dir=0 [E]
=1 [S]
=2 [W]
=3 [N]
=4 % 4=0

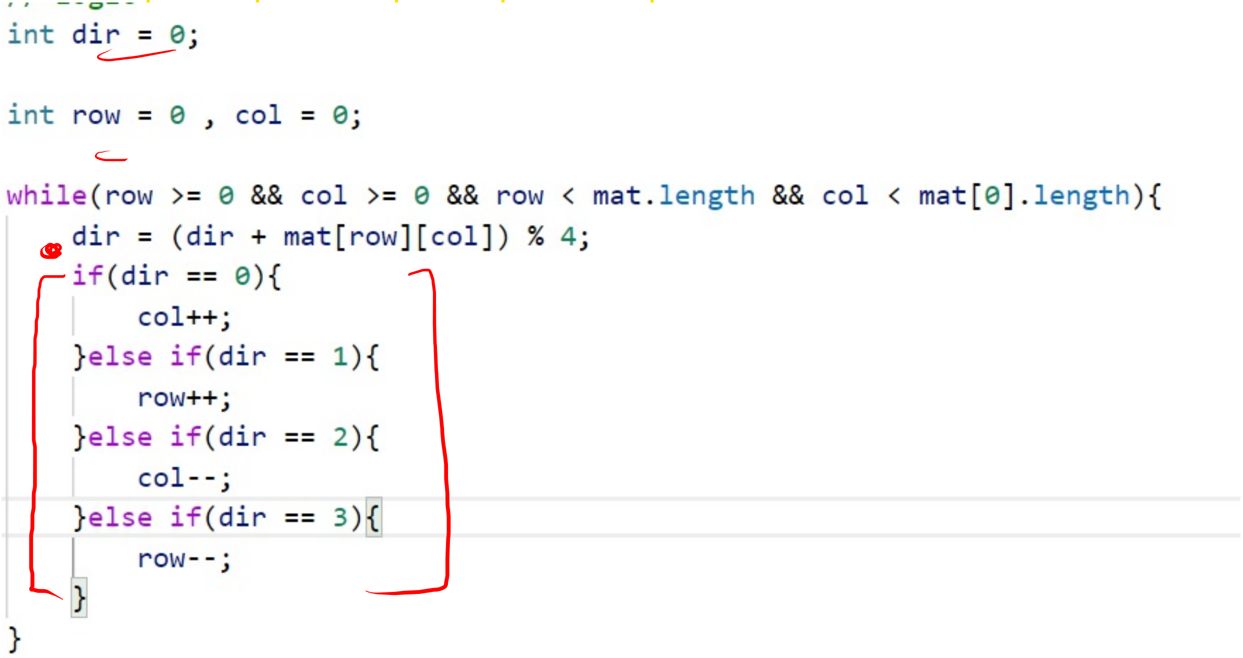




row	col	Dir
0	0	0
0	1	0
0	2	~~0~~ 1
1	2	1
2	2	1
3	2	2
3	1	2
3	0	3
2	0	3
		4 % 4=0
1	0	0
1	1	0
1	2	0
1	3	
	4	



	0	1	2	3
0	0	0	1	0
1	1	0	0	0
2	0	0	0	0
3	1	0	1	0



```java
// Logic
int dir = 0;

int row = 0 , col = 0;

while(row >= 0 && col >= 0 && row < mat.length && col < mat[0].length){
    dir = (dir + mat[row][col]) % 4;
    if(dir == 0){
        col++;
    }else if(dir == 1){
        row++;
    }else if(dir == 2){
        col--;
    }else if(dir == 3){
        row--;
    }
}
```

row	col	dir
-1	-1	
0	0	0 to ⇒0
0	1	0
pR 0	pC 2	1
R 1	C 2	1
2	2	1
3	2	2
3	1	2
3	0	3
2	0	3
1	0	0
1	1	0
1	2	0
pxvr → 1	pxvc → 3	0
ro → 1	col → 4	