ROTEIRO

1. Vetores Bidimensionais (Matrizes)
2. Vetores Ndimensionais

PROBLEMÁTICA

- Imagine o problema de se construir uma matriz origem/destino para a logística de produtos de 5 fábricas (A, B, C, D, E) para 10 centro de distribuições (1, 2, 3, 4, 5, 6, 7, 8, 9, 10)

	1	2	3	4	5	6	7	8	9	10	Soma
A	20			20				60			100
B	10	30	60						5	5	110
C					50		25		10	15	100
D						50	20				70
E				40	20			30			90
	30	30	60	60	70	50	45	90	15	20	Soma

PROBLEMÁTICA

- Podemos construir essa matriz utilizando cinco vetores
- int a[10], int b[10], int c[10], int d[10], int e[10]
- Mas e se quisermos trabalhar com x fábricas e y centro de distribuições?

	1	2	3	4	5	6	7	8	9	10	Soma
A	20			20				60			100
B	10	30	60						5	5	110
C					50		25		10	15	100
D						50	20				70
E				40	20			30			90
	30	30	60	60	70	50	45	90	15	20	Soma

VETORES BIDIMENSIONAIS (MATRIZES)

- Vetor é um ***array de uma dimensão.***
- A linguagem possibilita trabalhar com vetores com mais **dimensões**.
- Uma matriz é um vetor (array) de duas dimensões
- **Dimensões devem possuir o mesmo tipo de dado!!**
- Declaração:

  tipo_dado *nome_matriz*[dim1] [dim2];

Exemplo:

```
int sudoku[9] [9];
char cruzadas[5] [10];
float k[5] [10];
```

VETORES BIDIMENSIONAIS (MATRIZES)

- Declaração:

  tipo_dado *nome_matriz*[dim1] [dim2];

Exemplo de Inicialização:

```
int a[2] [2] = { 0 };
int m[2] [2] = { {1, 2}, {10, 20} };
```

a

	0	1
0	0	0
1	0	0

m

	0	1
0	1	2
1	10	20

VETORES BIDIMENSIONAIS (MATRIZES)

```
int M[2][3] = {
 { 1,  9, -13},  /* 1a linha */
 {20, -5,   6},  /* 2a linha */
};
```

$M_{2,3} =$

1	9	-13
20	-5	6

- Usamos indexação dupla *M*[i] [j]; para acessar elementos de M
  - i indexa na primeira dimensão (linha)
  - j indexa na segunda dimensão (coluna)
- *M*[i] [j] válido sse $0 \leq i < n$ e $0 \leq j < m$, para matriz *M*[n] [m]

- `int M[2][3];`
- `M[0][1] = 15;`
- `M[1][0] = M[0][1] * 2; /* qual o valor de M[1][0]? */`

VETORES BIDIMENSIONAIS (MATRIZES)

Exemplo de Inicialização:

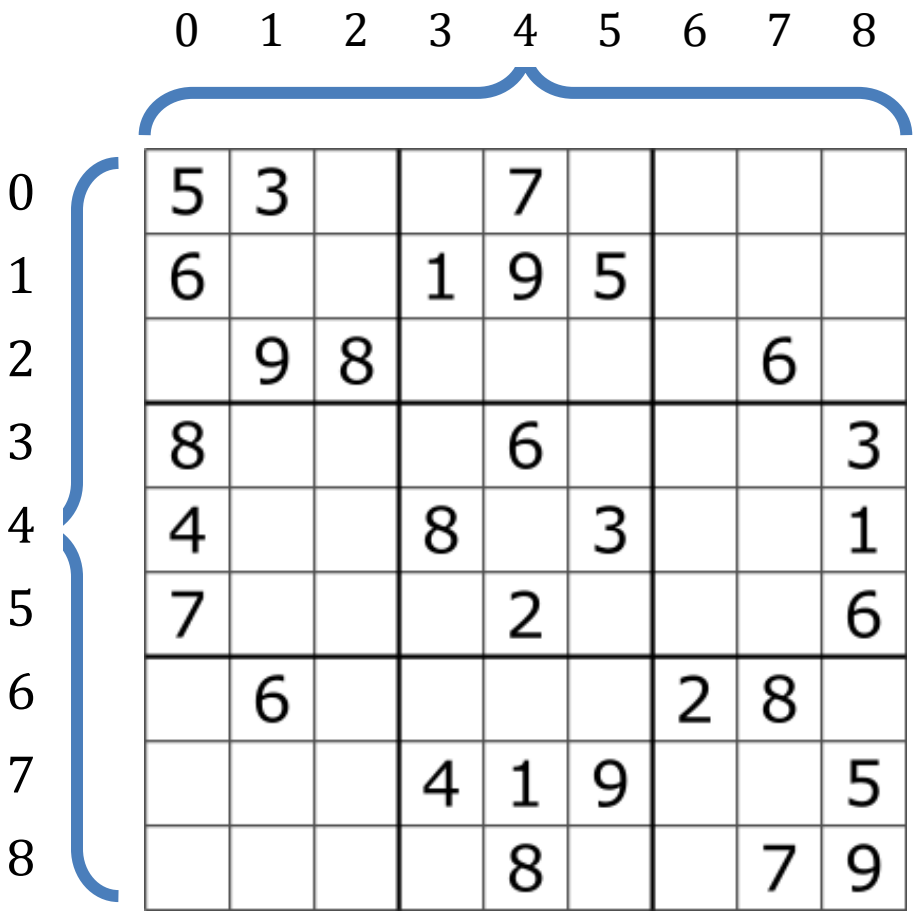
```
int a[2] [2] = { 0 };
int m[2] [2] = { {1, 2}, {10, 20} };
```

```c
#include <stdio.h>

int main() {
    int a[2] [2] = { 0 };
    int m[2] [2] = { {1, 2}, {10, 20} };

    printf("%d\n", a[0][1]);
    printf("%d\n", m[0][1]);
}
```

MATRIZES (VETORES BIDIMENSIONAIS)

- int *sudoku*[9] [9] = {0}:

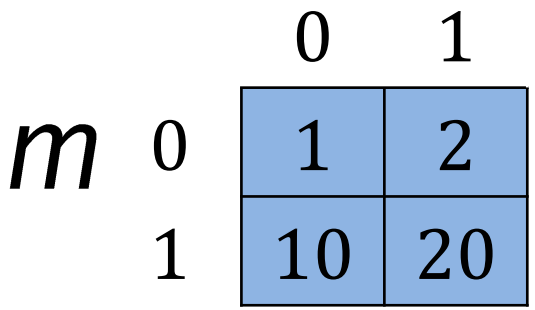
- sudoku[0][0] = 5;
- sudoku[0][1] = 3;
- sudoku[0][4] = 7;
- …

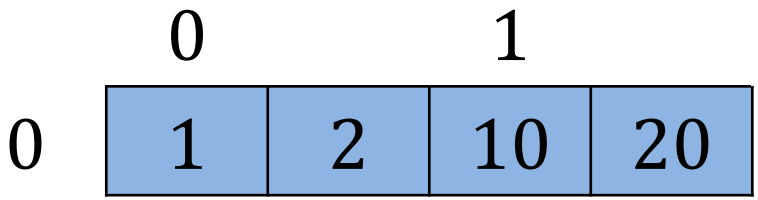


VETORES BIDIMENSIONAIS (MATRIZES)

- Os índices de uma matriz são lógicos
- Dados são armazenados como um vetor

```
int m[2][2] = { {1, 2}, {10, 20} };
```





VETORES BIDIMENSIONAIS (MATRIZES)

- Os índices de uma matriz são lógicos
- Dados são armazenados como um vetor

```
int m[3][3] = { 1, 2, 3, 10, 20, 30, 40, 20, 50 };
```

	0	1	2
0	1	2	3
1	10	20	30
2	40	20	50

	0			1			2		
0	1	2	3	10	20	30	40	20	50

VETORES BIDIMENSIONAIS (MATRIZES)

- Os índices de uma matriz são lógicos
- Dados são armazenados como um vetor

```
int m[][3] = { 1, 2, 3, 10, 20, 30, 40, 20, 50 };
```

	0	1	2
0	1	2	3
1	10	20	30
2	40	20	50

```
int m[][1]
```

	0
0	1
1	2
2	3
...	...
9	50

MATRIZES (EXEMPLO)

- `int od [5] [10];`
- `od [2] [4] = 30;`

	1	2	3	4	5	6	7	8	9	10	Soma
A	20			20				60			100
B	10	30	60						5	5	110
C					50		25		10	15	100
D						50	20				70
E				40	20			30			90
	30	30	60	60	70	50	45	90	15	20	Soma

MATRIZES (EXEMPLO)

- int *od* [5] [10];

```
1   #include <stdio.h>
2
3   int main() {
4       int od[5][10] = {
5                         {20,0,0,20,0,0,0,60,0,0},
6                         {10,30,60,0,0,0,0,0,5,5},
7                         {0,0,0,0,50,0,25,0,10,15},
8                         {0,0,0,40,20,0,0,30,0,0}
9                       };
10      printf("%d", od[0][3]);
11  }
12
```

LENDO UMA MATRIZ

- Ler um elemento por vez
- Precisar de dois laços (loops)

```c
#include <stdio.h>

int main() {
  int a[2][2];
  int i, j;
  /*Leitura*/
  for (i = 0; i < 2; i++) {
    for (j = 0; j < 2; j++) {
        printf ("Matriz[%d][%d]: ", i, j);
        scanf ("%d", &a[i][j]);
    }
  }
}
```

ESCREVENDO UMA MATRIZ

- Imprimir um elemento por vez
- Precisar de dois laços (loops)

```
/* Escrita */
for (i = 0; i < 2; i++) {
    for (j = 0; j < 2; j++) {
        printf("%3d", a[i][j]);
    }
    printf("\n");
}
```

SOMA DE ELEMENTOS

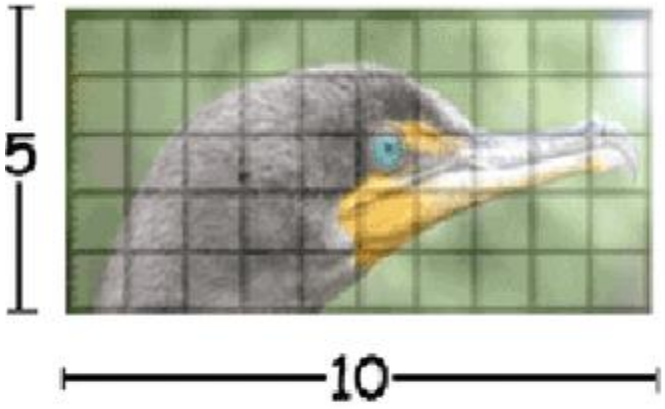
Operações em matrizes geralmente utilizam dois laços

Somar elementos

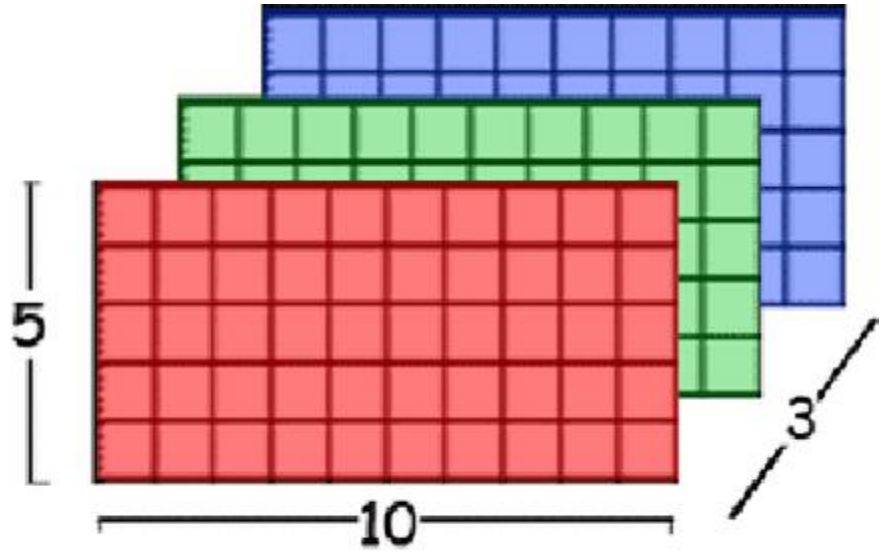
```
int i = 0;
int j = 0;
int soma = 0;
for (i = 0; i < 5; i++) {
    for (j = 0; j < 10; j++) {
        soma = soma + od[i][j];
    }
}

printf("Soma: %d", soma);
```

MATRIZES (NDIMENSIONAIS)

- int p[r] [g] [b];



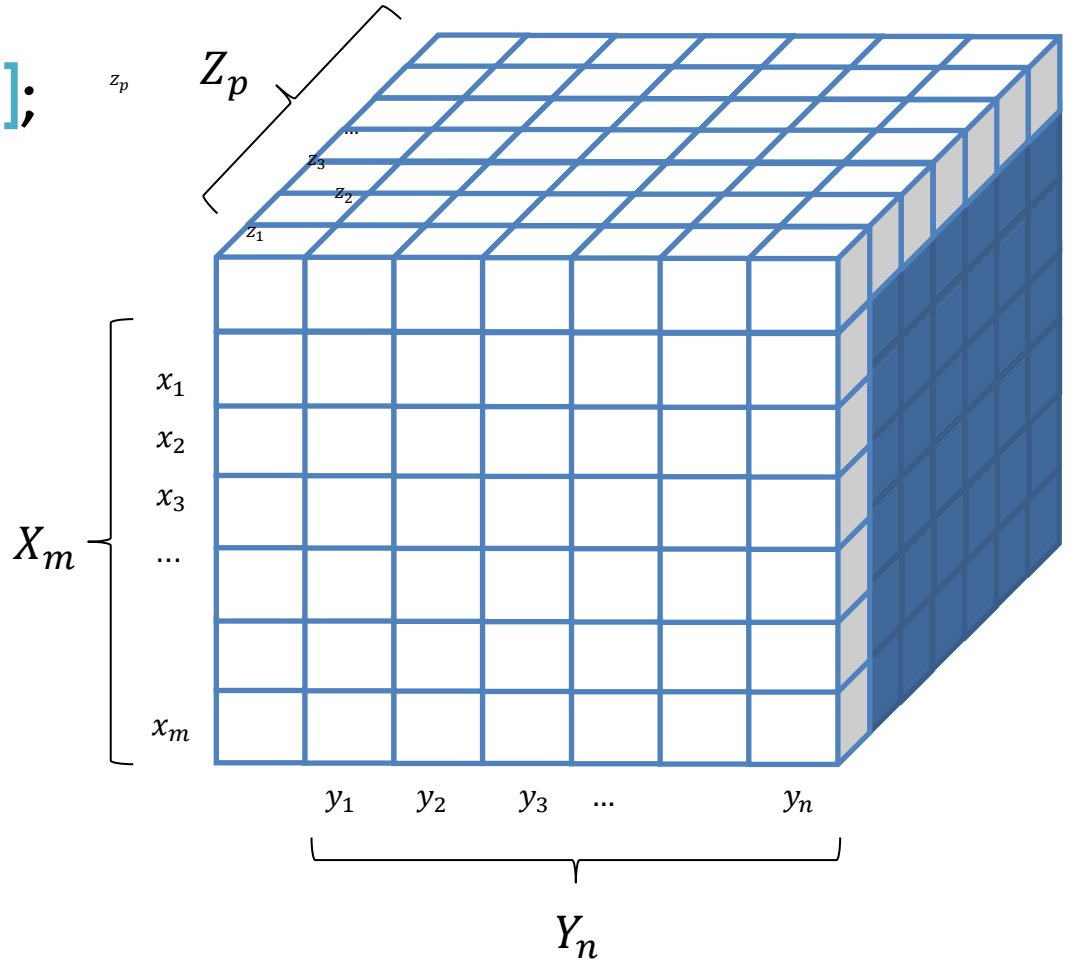
Original Color Image



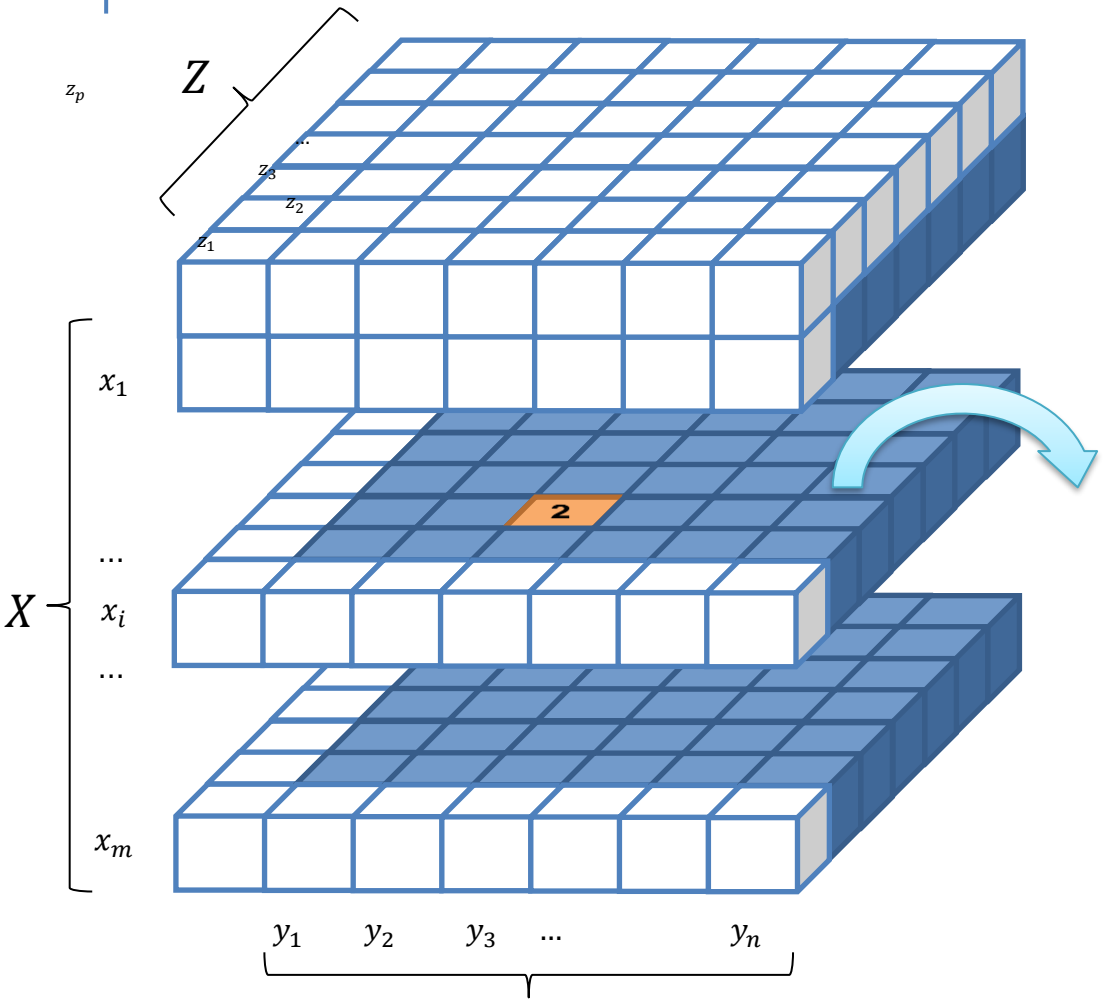
Matlab RGB Matrix

MATRIZES (NDIMENSIONAIS)

- int k[m] [n] [p];



MATRIZES (NDIMENSIONAIS)



- int k[m] [n] [p];

	z_1	z_2	z_3	...	...	z_k
	0	0	0	0	0	0	0
y_1	0	...	...	...	...	...	...
y_2	0	...	...	...	...	...	...
y_3	0	...	2	...	...	...	...
...	0	...	...	...	...	...	...
...	0	...	...	...	...	...	...
y_n	0	...	...	...	...	...	...

MATRIZES (NDIMENSIONAIS)

```
int v[3][4][2] = {
   {{1,3}, {-4,18}, {7,25}, {0,-3}},
   {{9,5}, {8,16}, {1,1}, {-8,7}},
   {{0,6}, {6,-3}, {8,15}, {12,0}},
};
```

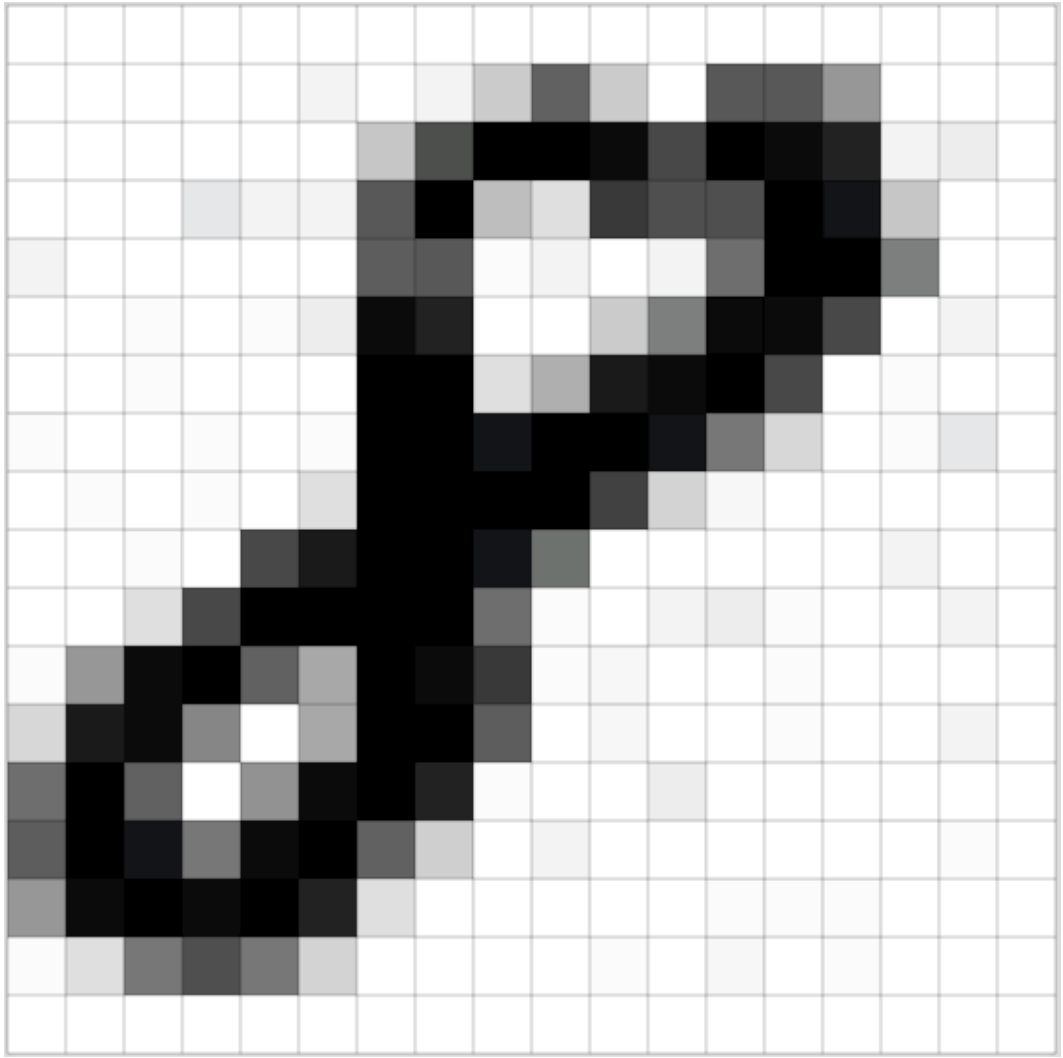
v é um vetor tridimensional com três elementos:
v[0], v[1], v[2], em que cada v[i] é uma matriz 4×2

- Qual o valor de v[0][3][1]?

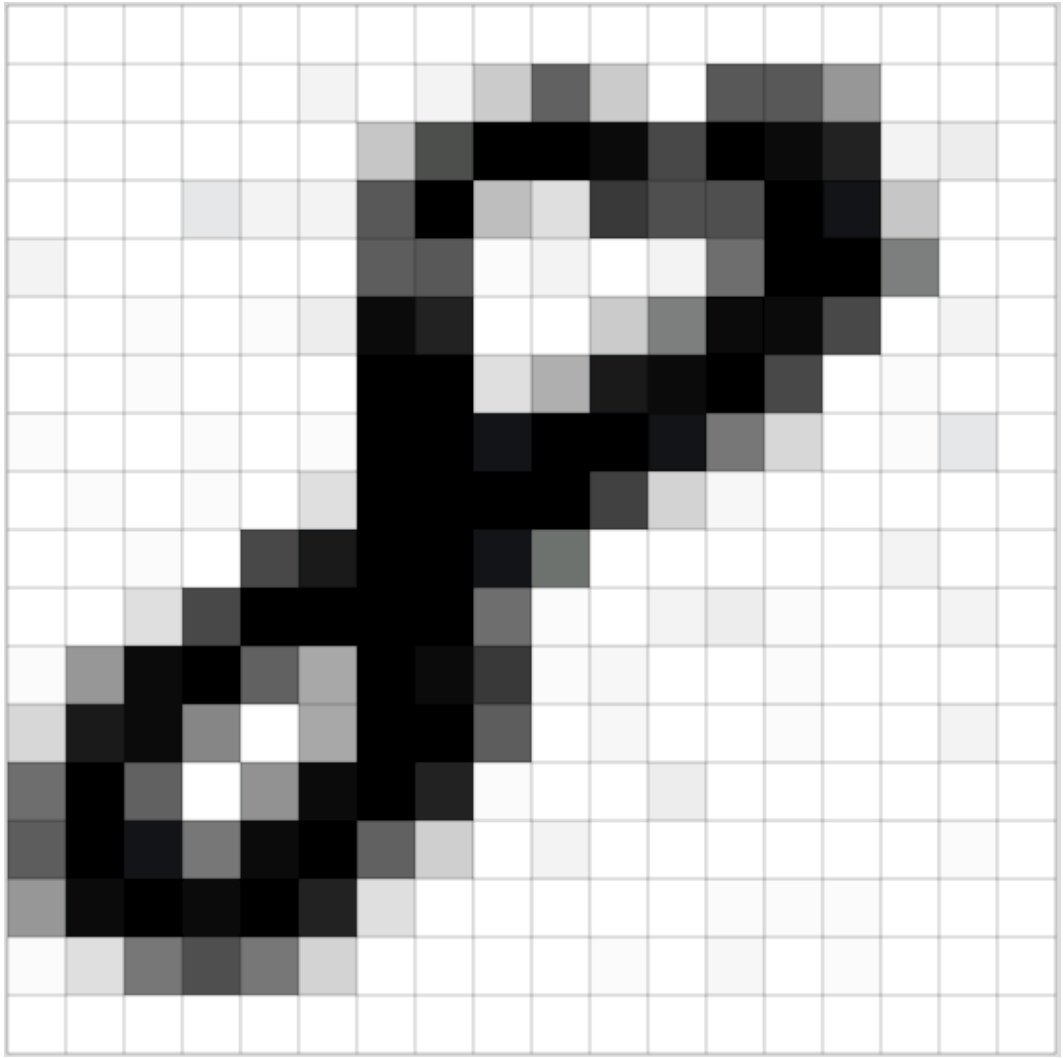
MATRIZES (APLICAÇÕES)

MATRIZES (APLICAÇÕES)

MATRIZES (APLICAÇÕES)



MATRIZES (APLICAÇÕES)



MATRIZES (APLICAÇÕES)

MATRIZES EXERCÍCIOS

- Escreva um programa que lê todos os elementos de uma matriz 5×5 e mostra a matriz e a sua transposta na tela.

Matriz

$$\begin{bmatrix} 0 & 1 & 0 & 2 & 3 \\ 0 & 1 & 0 & 2 & 3 \\ 0 & 1 & 0 & 2 & 3 \\ 0 & 1 & 0 & 2 & 3 \\ 0 & 1 & 0 & 2 & 3 \end{bmatrix}$$

Transposta

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 2 & 2 & 2 & 2 & 2 \\ 3 & 3 & 3 & 3 & 3 \end{bmatrix}$$

MATRIZES EXERCÍCIOS

- Escreva um programa que lê 2 matrizes 5 × 5, mostre-as na tela e mostre a soma entre as duas matrizes em seguida.

$$\underset{A}{\begin{bmatrix} 0 & 1 & 0 & 2 & 3 \\ 0 & 1 & 0 & 2 & 3 \\ 0 & 1 & 0 & 2 & 3 \\ 0 & 1 & 0 & 2 & 3 \\ 0 & 1 & 0 & 2 & 3 \end{bmatrix}} + \underset{B}{\begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 2 & 2 & 2 & 2 & 2 \\ 3 & 3 & 3 & 3 & 3 \end{bmatrix}} = \underset{C}{\begin{bmatrix} 0 & 1 & 0 & 4 & 3 \\ 1 & 2 & 1 & 3 & 4 \\ 0 & 1 & 0 & 2 & 3 \\ 2 & 3 & 2 & 4 & 5 \\ 3 & 4 & 3 & 5 & 6 \end{bmatrix}}$$

https://br.spoj.com/problems/ESCADA14/

ESCADA14 - Matriz Escada

Joãozinho está aprendendo sobre matrizes. Hoje ele aprendeu como deixar matrizes na forma escada, e está exercitando. Para ajudá-lo, você deve escrever um programa que determine se o resultado dele realmente está no formato correto.

Uma matriz está na forma escada quando, para cada linha, as condições a seguir forem satisfeitas:

• Se a linha só possuir zeros, então todas as linhas abaixo desta também só possuem zeros.
• Caso contrário, seja X o elemento diferente de zero mais à esquerda da linha; então, pa elementos nas colunas à esquerda de X e na coluna de X são iguais a zero.

Entrada

A primeira linha possui dois inteiros N e M , as dimensões da matriz. Cada uma das N linh elementos da matriz.

https://br.spoj.com/problems/MINHOCA/

MINHOCA - Campo de Minhocas

Minhocas são muito importantes para a agricultura e como insumo para produção de ração animal. A Organização para Bioengenharia de Minhocas (OBM) é uma entidade não governamental que promove o aumento da produção, utilização e exportação de minhocas.

Uma das atividades promovidas pela OBM é a manutenção de uma fazenda experimental para pesquisa de novas tecnologias de criação de minhocas. Na fazenda, a área destinada às pesquisas é de formato retangular, dividida em células quadrangulares de mesmo tamanho. As células são utilizadas para testar os efeitos, na produção de minhocas, de variações de espécies de minhocas, tipos de terra, de adu mantêm um acompanhamento constante do desenvolvimento das minhocas em cada cé produtividade em cada uma das células. A figura abaixo mostra um mapa da fazenda, m células.

81	28	240	10
40	10	100	240