Universidade Federal de Viçosa
Departamento de Informática
Centro de Ciências Exatas e Tecnológicas

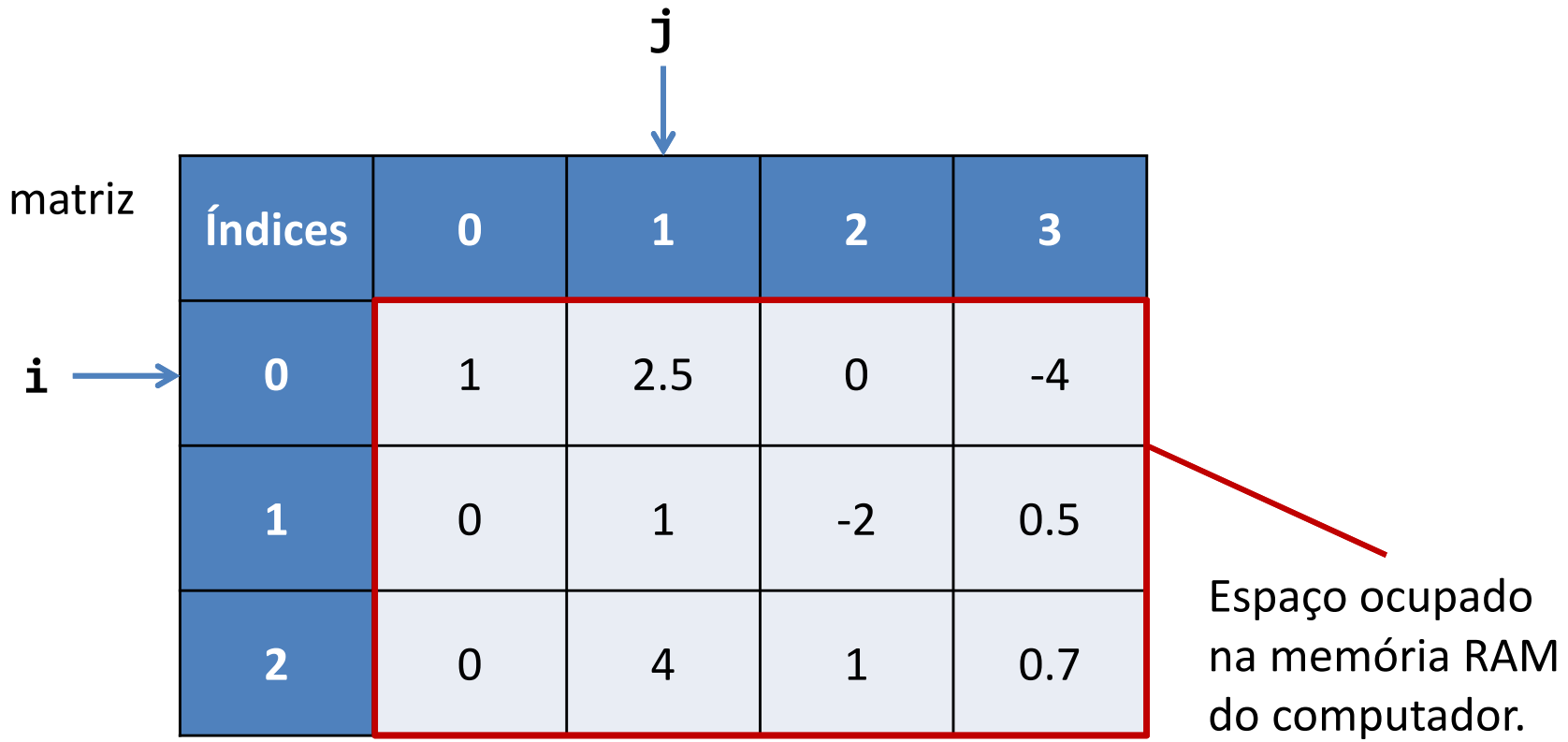
INF 100 – Introdução à Programação

Arranjos Simples 2D

(matrizes)

Arranjos Multidimensionais

```
matriz[i][j]  # ref. à linha i, coluna j da matriz
```



Percorrendo Matrizes

```
para cada linha i da matriz
    para cada coluna j da matriz
        faça alguma coisa com o elemento a[i][j]
```

Percorrendo Matrizes

```
para cada linha i da matriz
    para cada coluna j da matriz
        faça alguma coisa com o elemento a[i][j]
```

Este loop é executado PARA CADA valor de i

Percorrendo Matrizes

```
para cada linha i da matriz
    para cada coluna j da matriz
        faça alguma coisa com o elemento a[i][j]
```

Este(s) comando(s) é(são) executado(s) PARA CADA valor de i e j


Arranjos Multidimensionais

```
# Ler uma matriz elemento por elemento...

import numpy as np

A = np.empty( (3, 4) )  # matriz A: 3 x 4

for i in range( 0, 3 ): # linhas 0 a 2
    for j in range( 0, 4 ): # colunas 0 a 3
        s = 'Informe o elemento A[%d][%d]: ' % (i, j)
        A[i][j] = float (input( s ))
```


Arranjos Multidimensionais

```
# Ler uma matriz elemento por elemento...

import numpy as np

# Ler dimensões da matriz
m = int( input('Número de linhas : '))
n = int( input('Número de colunas: '))

# criar matriz A: m x n
A = np.empty( (m, n) )

# Ler matriz do teclado
for i in range( 0, m ): # linhas 0 a m-1
    for j in range( 0, n ): # colunas 0 a n-1
        s = 'Informe o elemento A[%d][%d]: ' % (i, j)
        A[i][j] = float (input( s ))
```

Arranjos Multidimensionais em Python

```
# Escrever uma matriz de inteiros na tela
# (formatada)

mi = np.array([[1, 2, 3],
               [4, 5, 6],
               [7, 8, 9],
               [10, 11, 12]])

for i in range(0, 4):
    for j in range(0, 3):
        print('%3d' % (mi[i][j]), end='')
    print()
```

Resultado:

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

Arranjos Multidimensionais em Python

```
# Escrever uma matriz de float na tela
# (formatada)

mf = np.array([[1.1, 2.5, 3],
               [4, 5.2, 6],
               [7.8, 8, 9.9],
               [10, 11, 12.3]])

for i in range(0, 4):
    for j in range(0, 3):
        print('%5.1f' % (mf[i][j]), end='')
    print()
```

Resultado:

```
  1.1  2.5  3.0
  4.0  5.2  6.0
  7.8  8.0  9.9
 10.0 11.0 12.3
```

Mais Exemplos de Formatação de Saída em Python

```
for i in range(0, 4):
    for j in range(0, 3):
        print('%7.2f' % (mf[i][j]), end='')
    print()
```

```
   1.10    2.50    3.00
   4.00    5.20    6.00
   7.80    8.00    9.90
  10.00   11.00   12.30
```

Mais Exemplos de Formatação de Saída em Python

```
for i in range(0, 4):
    for j in range(0, 3):
        print('%3d' % (mi[i][j]), end='')
    print()
```

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

Mais Exemplos de Formatação de Saída em Python

```
for i in range(0, 4):
    for j in range(0, 3):
        vij = int( mf[i][j] )
        vij10 = int( mf[i][j]*10 )
        print('%4d.%d' % (vij, vij10 % 10), end='')
    print()
```

```
   1.1   2.5   3.0
   4.0   5.2   6.0
   7.8   8.0   9.9
  10.0  11.0  12.3
```

Mais Exemplos de Formatação de Saída em Python

```
for i in range(0, 3):
    print('v[%d][%d] =%5.1f' % ( i, i, mf[i][i] ))
```

```
v[0][0] =  1.1
v[1][1] =  5.2
v[2][2] =  9.9
```

```
for i in range(0, 3):
    j = 2 - i
    print('v[%d][%d] =%5.1f' % ( i, j, mf[i][j] ))
```

```
v[0][2] =  3.0
v[1][1] =  5.2
v[2][0] =  7.8
```

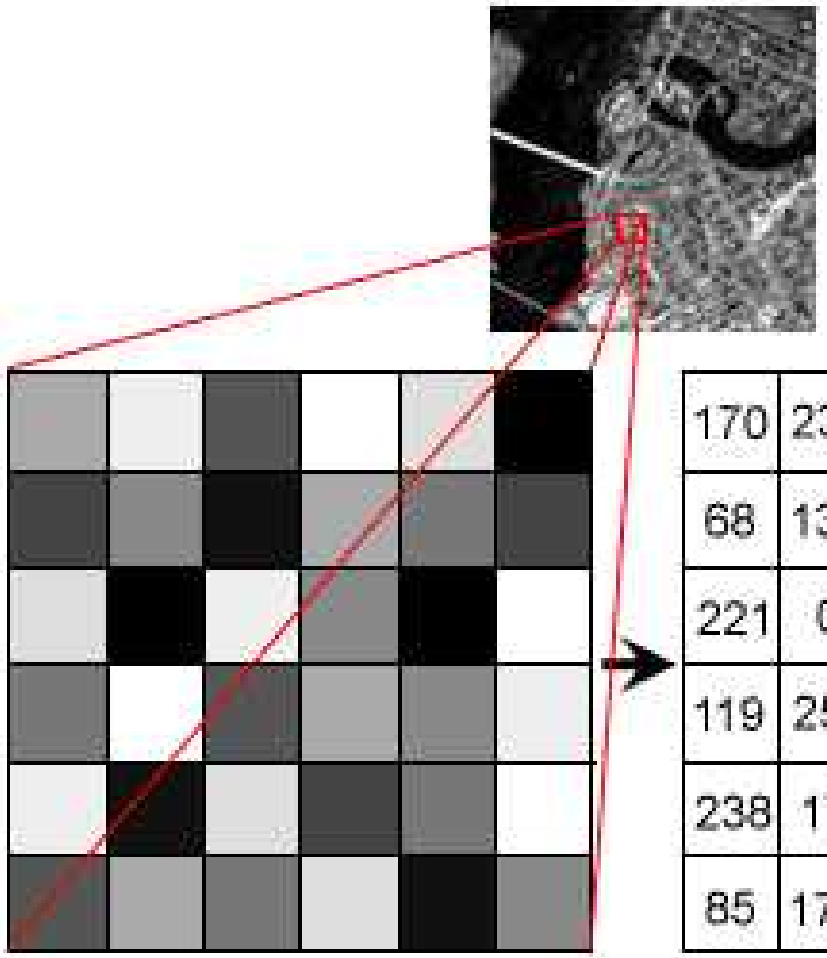

Exercício 1

- Faça um programa que leia os dados de uma matriz A de números reais, de dimensões *m* x *n*. Em seguida, ele deve:
  a) Preencher um vetor B, de dimensão *m*, onde cada uma de suas células contém a soma dos valores da linha correspondente da matriz A;
  b) Preencher um vetor C, de dimensão *n*, onde cada uma de suas células contém a soma dos valores da coluna correspondente da matriz A;

Exercício 1

```
import numpy as np

m = int( input('Número de linhas : '))
n = int( input('Número de colunas: '))
A = np.empty( (m, n) )
for i in range(0, m): # linhas 0 a m-1
    for j in range(0, n): # colunas 0 a n-1
        s = 'Informe o elemento A[%d][%d]: ' % ( i, j )
        A[i][j] = float (input( s ))

print('\nMatriz lida:\n')
for i in range(0, m):
    for j in range(0, n):
        print('%6.1f' % ( A[i][j] ), end='')
    print()

B = np.empty( m )
for i in range(0, m):
    for j in range(0, n):
        B[i] = B[i] + A[i][j]

C = np.empty( n )
for j in range(0, n):
    for i in range(0, m):
        C[j] = C[j] + A[i][j]

# Escreve B
print('\nB = ', end='')
for i in range(0, m):
    print('%6.1f' % ( B[i] ), end='')
print()

# Escreve C
print('\nC = ', end='')
for i in range(0, n):
    print('%6.1f' % ( C[i] ), end='')
print()
```

Imagens Digitais



170	238	85	255	221	0
68	136	17	170	119	68
221	0	238	136	0	255
119	255	85	170	136	238
238	17	221	68	119	255
85	170	119	221	17	136

1 pixel

Imagem em tons de cinza:
256 tons de cinza
0 = preto
255 = branco

Imagens Digitais

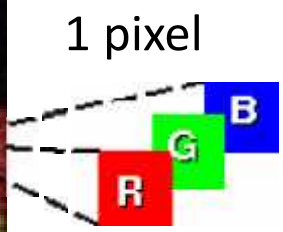
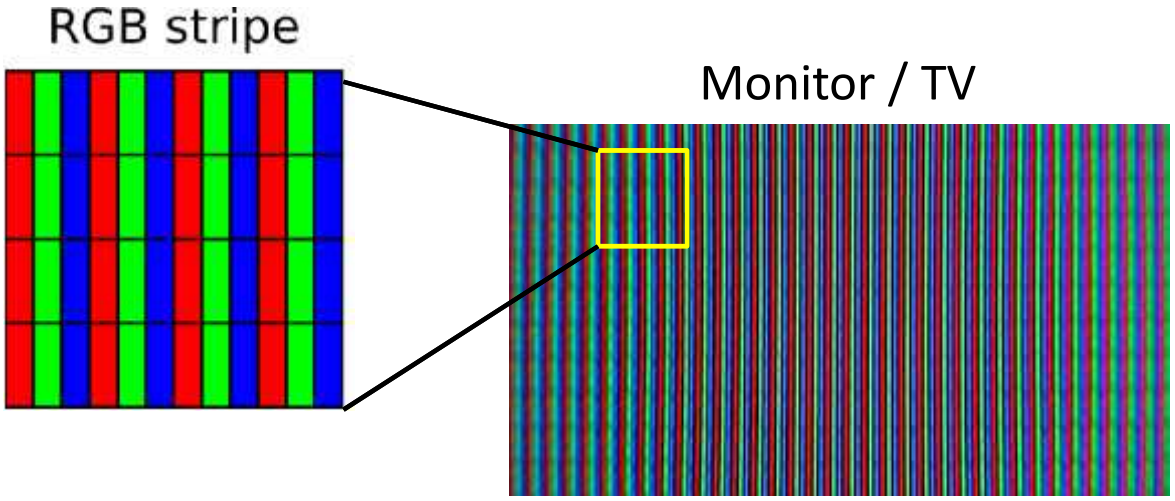


Imagem colorida:
Cada componente (RGB) do pixel é um valor entre 0 e 255

$256^3 \approx 16$ milhões de cores



Processamento Digital de Imagens

Algoritmo Genérico

```
Para cada ponto (pixel) da imagem:
    Ler o valor do pixel da imagem
    Alterar o valores (cor ou cores) do pixel
    Escrever o valor do pixel na imagem
fim_para
```

Processamento Digital de Imagens

Algoritmo Genérico

```
largura = pegar largura da imagem
altura = pegar altura da imagem

para y = 0 até altura-1:
    para x = 0 até largura-1:
        pixel = imagem[y][x]
        Alterar valor do pixel
        imagem[y][x] = pixel
```

Exercício 2

- Faça um programa que:
  a) Leia uma imagem digital, exibindo-a na tela;
  b) Altere o valor dos pixels, aumentando o componente verde em 50% e o azul em 30%, e reduzindo o vermelho em 10%;
  c) Exiba a imagem resultante.

Exercício 2

```
import imagens

im = imagens.Imagem('lenna_original_RGB.jpg')
im.mostrar()  # Mostrar a imagem na tela

# Produz imagem negativa
for y in range( 0, im.altura ):
    for x in range( 0, im.largura ):

        r, g, b = im[y][x]



        r = int( r * 0.9 )  # reduzir o vermelho em 10%
        g = min( 255, int( g * 1.5 ))  # aumentar o verde em 50%
        b = min( 255, int( b * 1.3 ))  # aumentar o azul em 30%

        im[y][x] = (r,g,b)
im.mostrar()
```

Em imagens coloridas, cada pixel é composto por 3 componentes:
- r (red, vermelho)
- g (green, verde)
- b (blue, azul)

Cada componente é um valor inteiro 0..255

Exercício 3

- Faça um programa que leia os dados de uma matriz quadrada A, de dimensões *m* x *m*. Em seguida, deve fazer as seguintes operações:
  a) Exibir em tela o produto dos valores da diagonal principal;
  b) Exibir em tela o produto dos valores da diagonal secundária;
  c) Calcule a média dos valores contidos na matriz e informe quantos deles estão abaixo da média.


Exercício 3

```python
import numpy as np

m = int( input('m = '))
A = np.empty( (m, m) )
for i in range(0, m): # linhas 0 a m-1
    for j in range(0, m): # colunas 0 a m-1
        s = 'Informe o elemento A[%d][%d]: ' % ( i, j )
        A[i][j] = float (input( s ))

print('\nMatriz lida:\n')
for i in range(0, m):
    for j in range(0, m):
        print('%6.1f' % ( A[i][j] ), end='')
    print()

p = 1
for i in range(0, m):
    p = p * A[i][i]
print('\nProduto dos elementos da diagonal principal =', p )

p = 1
for i in range(0, m):
    p = p * A[i][m-1-i]
print('Produto dos elementos da diagonal secundária =', p )

# Calculando a média...
soma = 0
for i in range(0, m):
    for j in range(0, m):
        soma = soma + A[i][j]
media = soma/(m*m)
print('Média dos elementos da matriz:', media )

# Nº elementos abaixo da média...
nAbaixo = 0
for i in range(0, m):
    for j in range(0, m):
        if A[i][j] < media: nAbaixo = nAbaixo + 1
print('Nº de elementos abaixo da média:', nAbaixo )
```

Exercício 4

- Faça um programa que leia os dados de duas matrizes de números inteiros A_{mn} e B_{op} e exiba em tela os dados de uma matriz C_{qr}, resultado do produto de A por B, caso este produto seja possível/viável.


Exercício 4

```
import numpy as np
import sys

print('Faz a multiplicação da matriz A: m x n pela matriz
B: o x p\n')

m = int( input('m = '))
n = int( input('n = '))
o = int( input('o = '))
p = int( input('p = '))

if n != o:
    # A função sys.exit força o término do programa,
    # mostrando uma mensagem na tela
    sys.exit('Não é possível multiplicar essas matrizes!')

A = np.empty( (m, n) )
print('\nEntre com a matriz A (m x n):')
for i in range(0, m): # linhas 0 a m-1
    for j in range(0, n): # colunas 0 a n-1
        s = 'A[%d][%d] = ' % ( i, j )
        A[i][j] = float (input( s ))

B = np.empty( (n, p) )
print('\nEntre com a matriz B (o x p):')
for i in range(0, n): # linhas 0 a n-1
    for j in range(0, p): # colunas 0 a p-1
        s = 'B[%d][%d] = ' % ( i, j )
        B[i][j] = float (input( s ))

# Calculando o produto...
C = np.empty( (m, p) )
for i in range(0, m):
    for j in range(0, p):
        soma = 0
        for k in range(0, n):
            soma = soma + A[i][k] * B[k][j]
        C[i][j] = soma

# Escreve as matrizes A, B e C na tela...

print('\nMatriz A:\n')
for i in range(0, m):
    for j in range(0, n):
        print('%6.1f' % ( A[i][j] ), end='')
    print()

print('\nMatriz B:\n')
for i in range(0, n):
    for j in range(0, p):
        print('%6.1f' % ( B[i][j] ), end='')
    print()

print('\nMatriz C = A x B:\n')
for i in range(0, m):
    for j in range(0, p):
        print('%6.1f' % ( C[i][j] ), end='')
    print()
```

Exercício 5

- No jogo Campo Minado, um certo número de bombas é distribuído em um campo, representado por uma matriz. Uma célula que não contém uma bomba deve indicar o número de bombas existentes em volta dela.
- Faça um programa que lê a dimensão da matriz e o número de bombas. Depois ele deve distribuir as bombas na matriz aleatoriamente e então calcular o número de bombas existentes em torno de cada célula 'não bomba'.


Exercício 5

Exemplo de execução

- Obs.: suponha que as bombas são representadas pelo valor 9.

```
Tamanho do campo minado: 4
Número de bombas: 5

Distribuindo as bombas:
 0 0 0 0
 0 9 9 0
 9 9 0 0
 0 0 9 0

Calculando as bombas em torno das células:
 1 2 2 1
 3 9 9 1
 9 9 4 2
 2 3 9 1
```

Exercício 5

```
import numpy as np
import random

bomba = 9

n = int(input('Tamanho do lado do campo minado: '))
while True:
    b = int(input('Número de bombas: '))
    if b > n*n//2:
        print('Você colocou bombas demais!')
    else: break

campo = np.zeros((n, n), dtype=int)

k = b
while k > 0:
    i = random.randint(0, n-1)
    j = random.randint(0, n-1)
    if campo[i][j] == 0:
        campo[i][j] = bomba
        k = k - 1

print()
for i in range(0, n):
    for j in range(0, n):
        print('%2d' % (campo[i][j]), end='')
    print()

for i in range(0, n):
    for j in range(0, n):
        if campo[i][j] == bomba:
            if i > 0 and j > 0 and campo[i-1][j-1] != bomba:
                campo[i-1][j-1] = campo[i-1][j-1] + 1
            if i > 0 and campo[i-1][j] != bomba:
                campo[i-1][j] = campo[i-1][j] + 1
            if i > 0 and j < n-1 and campo[i-1][j+1] != bomba:
                campo[i-1][j+1] = campo[i-1][j+1] + 1
            if j > 0 and campo[i][j-1] != bomba:
                campo[i][j-1] = campo[i][j-1] + 1
            if j < n-1 and campo[i][j+1] != bomba:
                campo[i][j+1] = campo[i][j+1] + 1
            if i < n-1 and j > 0 and campo[i+1][j-1] != bomba:
                campo[i+1][j-1] = campo[i+1][j-1] + 1
            if i < n-1 and campo[i+1][j] != bomba:
                campo[i+1][j] = campo[i+1][j] + 1
            if i < n-1 and j < n-1 and campo[i+1][j+1] != bomba:
                campo[i+1][j+1] = campo[i+1][j+1] + 1

print()
for i in range(0, n):
    for j in range(0, n):
        print('%2d' % (campo[i][j]), end='')
    print()
```

Exercício 6

- Faça um programa que implementa o seguinte algoritmo de alto nível:
  1. Ler um valor inteiro n.
  2. Ler do teclado ou gerar aleatoriamente uma matriz A de dimensões n x n de números inteiros.
  3. Escrever a matriz A na tela.
  4. Fazer $A \leftarrow A^T$
  5. Escrever a matriz A na tela.

Exercício 6

```python
import random
import numpy as np

n = int( input('Entre com a ordem da matriz: '))

random.seed()

# gerar matriz A: n x n de valores inteiros aleatórios entre 1 e 99
A = np.empty( (n, n), dtype=int )
for i in range(0, n):
    for j in range(0, n):
        A[i][j] = random.randint( 1, 99 )

# Escrever a matriz A na tela
print('\nMatriz A:')
for i in range(0, n):
    for j in range(0, n):
        print('%4d' % ( A[i][j] ), end='')
    print()

# Transpor a matriz A
print('\nTranspondo matriz A...')
for i in range(0, n):
    for j in range(i+1, n):
        A[i][j], A[j][i] = A[j][i], A[i][j]

# Escrever a matriz A na tela
print('\nMatriz A:')
for i in range(0, n):
    for j in range(0, n):
        print('%4d' % ( A[i][j] ), end='')
    print()
```

Exercício 7

- Faça um programa que implementa o seguinte algoritmo de alto nível:
  1. Ler os valores inteiros m e n.
  2. Ler do teclado ou gerar aleatoriamente uma matriz A de dimensões m x n de números reais.
  3. Escrever a matriz A na tela.
  4. Gerar a matriz B como sendo a transposta de A.
  5. Escrever a matriz B na tela.


Exercício 7

```
import random
import numpy as np

m = int( input('Entre com o número de linhas da matriz:  '))
n = int( input('Entre com o número de colunas da matriz: '))

random.seed()

# gerar matriz A: n x n de valores reais aleatórios entre 1 e 99
A = np.empty( (m, n) )
for i in range(0, m):
    for j in range(0, n):
        A[i][j] = random.uniform( 1, 99 )

# Escrever a matriz A na tela
print('\nMatriz A:')
for i in range(0, m):
    for j in range(0, n):
        print('%4.0f' % ( A[i][j] ), end='')
    print()

# Transpor a matriz A
print('\nTranspondo matriz A...')
B = np.empty( (n, m) )
for i in range(0, m):
    for j in range(0, n):
        B[j][i] = A[i][j]

# Escrever a matriz A na tela
print('\nMatriz B:')
for i in range(0, n):
    for j in range(0, m):
        print('%4.0f' % ( B[i][j] ), end='')
    print()
```