



## 503202/503203 Programación Laboratorio de Programación Usando Arreglos

### EQUIPO PROGRAMACIÓN

15 de junio de 2025

- 1.- Construya un programa en Python que lea un conjunto de 12 números reales (punto flotante), los cuales deben ser almacenados en un vector  $C$ . Luego el programa debe desplegar los valores en orden inverso al usado para ingresarlos. Por ejemplo, si se ingresa los valores:

1.2 1.4 5.6 -7.8 9.6 -0.2 -9.3 3.3 4.4 1.9 -8.5 3.5

el programa debe desplegar

3.5 -8.5 1.9 4.4 3.3 -9.3 -0.2 9.6 -7.8 5.6 1.4 1.2

- 2.- Construya un programa en Python que lea un conjunto de 16 números reales (punto flotante), los cuales deben ser almacenados en un vector  $C$ . Luego el programa debe desplegar las sumas de las dos mitades del arreglo. Por ejemplo, si se ingresa los valores:

1.2 1.4 5.6 -7.8 9.6 -0.2 -9.3 3.3 4.4 1.9 -8.5 3.5 3.6 2.8 4.4 -1.2

el programa debe desplegar los valores 3.8 10.9

- 3.- Escriba un programa en Python que lea dos matrices  $A$  y  $B$  de  $n \times n$ ,  $n$  debe ser leído, y que calcule una matriz  $C$  que sea la suma de  $A$  con  $B$  (no use funciones).

Entradas: las entradas del programa serán el valor de  $n$ , ( $1 < n \leq 10$ ) y luego vendrán  $n \times n$  valores correspondientes a la matriz  $A$  y  $n \times n$  valores correspondientes a la matriz  $B$ .

Salida: Una matriz de  $n \times n$  valores cumpliendo especificación de enunciado.

Ejemplo de entrada: 3

$$\begin{bmatrix} 2 & 7 & 6 \\ 4 & 5 & 3 \\ 9 & 8 & 1 \end{bmatrix} \quad (1)$$

$$\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \quad (2)$$

Ejemplo de salida:

$$\begin{bmatrix} 3 & 8 & 7 \\ 5 & 6 & 4 \\ 10 & 9 & 2 \end{bmatrix} \quad (3)$$

- 
- 4.- Escriba un programa en Python que lea dos matrices  $A$  y  $B$  de  $n \times n$ ,  $n$  debe ser leído, y que calcule una matriz  $C$  que sea el producto de  $A$  con  $B$  (no use funciones).

Entradas: las entradas del programa serán el valor de  $n$ , ( $1 < n \leq 10$ ) y luego vendrán  $n \times n$  valores correspondientes a la matriz  $A$  y  $n \times n$  valores correspondientes a la matriz  $B$ .

Salida: Una matriz de  $n \times n$  valores cumpliendo especificación de enunciado.

Ejemplo de entrada: 3

$$\begin{bmatrix} 2 & 7 & 6 \\ 4 & 5 & 3 \\ 9 & 8 & 1 \end{bmatrix} \quad (4)$$

$$\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \quad (5)$$

Ejemplo de salida:

$$\begin{bmatrix} 15 & 15 & 15 \\ 12 & 12 & 12 \\ 18 & 18 & 18 \end{bmatrix} \quad (6)$$

- 5.- Uno de los pasos que se requieren para resolver un sistema de ecuaciones lineales consiste en intercambiar las filas de una matriz cuadrada de manera de colocar en su diagonal principal los elementos de mayor valor en cada columna. Escriba un programa en Python que lea una matriz de  $n \times n$ ,  $n$  debe ser leído, y que intercambie las filas desde arriba hacia abajo de manera que el elemento de mayor magnitud de cada columna se ubique en la diagonal principal y sustituya con ceros el resto de la fila hacia la derecha.

Entradas: las entradas del programa serán el valor de  $n$ , ( $1 < n \leq 10$ ) y luego vendrán  $n \times n$  valores de la matriz.

Salidas: Una matriz de  $n \times n$  valores cumpliendo especificación de enunciado.

Ejemplo de entrada: 3

$$\begin{bmatrix} 2 & 7 & 6 \\ 4 & 5 & 3 \\ 9 & 8 & 1 \end{bmatrix} \quad (7)$$

Ejemplo de salidas: Una matriz de  $n \times n$  valores cumpliendo especificación de enunciado.

$$\begin{bmatrix} 9 & 0 & 0 \\ 2 & 7 & 0 \\ 4 & 5 & 3 \end{bmatrix} \quad (8)$$

Observación: Los pasos intermedios son:

$$\begin{bmatrix} 2 & 7 & 6 \\ 4 & 5 & 3 \\ 9 & 8 & 1 \end{bmatrix} \quad (9)$$

Se intercambia la fila 3 (por que 9 es mayor en columna 1) con fila 1

$$\begin{bmatrix} 9 & 0 & 0 \\ 4 & 5 & 3 \\ 2 & 7 & 6 \end{bmatrix} \quad (10)$$

---

Se intercambia la fila 3 (por que 7 es mayor en columna 2) con fila 2

$$\begin{bmatrix} 9 & 0 & 0 \\ 2 & 7 & 0 \\ 4 & 5 & 3 \end{bmatrix} \quad (11)$$

- 6.- Hay una guerra entre abejas y hormigas. Las abejas deciden tomar la delantera y desean lanzar muchas bombas de azúcar en el territorio de las hormigas. Supongamos que las hormigas están en una formación que simula un tablero de  $n$  filas por  $n$  columnas.

Se le pide implementar un programa en Python que contenga lo siguiente: una función que reciba dos parámetros. El primer parámetro corresponde a las dimensiones del tablero ( $n \times n$ ), el segundo parámetro corresponde al número de bombas que dispararan las abejas; un programa principal donde se muestre que inicialmente ninguna casilla tiene azúcar, es decir, cada casilla tiene valor cero y que cuando una bomba de azúcar cae en una casilla (seleccionada por teclado) el valor de azúcar de esa casilla aumenta en uno. Además, el azúcar se esparce a sus alrededores, por lo que cada uno de sus cuatro vecinos directos (arriba, abajo, izquierda y derecha) también incrementarán su monto de azúcar en uno; finalmente, la función debe retornar el tablero generado tras el bombardeo de azúcar en cada casilla.

Entradas: Este programa tiene  $2+2m$  entradas:  $n$ , el número de filas y columnas en la formación de las hormigas;  $m$ , el número de bombas de azúcar disparadas por las abejas, ambos deben ser valores positivos. Y  $m$  pares de números enteros representando las coordenadas de impacto de las bombas de azúcar lanzadas por las abejas.

Salida: La salida de este programa es el arreglo/la matriz que representa el estado de las hormigas tras el bombardeo de las abejas.

- 7.- Cree un programa en Python que lea los precios de 5 artículos y las cantidades vendidas por una empresa en sus 4 sucursales. Almacene las cantidades vendidas en una matriz de  $5 \times 4$ , donde las 5 filas representan las cantidades vendidas de cada uno de los 5 productos y las 4 columnas representan las 4 sucursales. Luego despliegue:

- Cantidad de artículos vendidos en la sucursal 2
- Cantidad de artículos vendidos en la sucursal 3
- Cantidad vendida del artículo 3 en la sucursal 1
- Recaudación total de la sucursal 4
- Recaudación total de la empresa

Entradas: en la primera línea vendrán 5 valores, que son los precios de cada producto. Luego la matriz que representa la cantidad de productos vendidos por cada sucursal.

Salidas: son los elementos a desplegar.

Ejemplo de entrada:

```
1000 990 1390 1500 2500
10 9 8 7 3 4 5 6 1 0 1 1 10 12 12 12 5 5 10 6
```

Ejemplo de salida: Matriz de cantidades vendidas:

- Cantidad de artículos vendidos en la sucursal 2: 30
- Cantidad de artículos vendidos en la sucursal 3: 36

- 
- Cantidad vendida del artículo 3 en la sucursal 1: 1
  - Recaudación total de la sucursal 4: 47330
  - Recaudación total de la empresa: 189990

8.- Desarrolle un programa Python que lea una matriz  $n \times n$  y determine si es o no simétrica. Desarrolle un programa similar para determinar si una matriz es o no hermitiana.

Nota: Una matriz simétrica es aquella que es igual a su traspuesta.

Una matriz hermitiana es aquella que es igual a la conjugada de su traspuesta.

(ver <https://www.matricesydeterminantes.com/matrices/tipos-de-matrices/matriz-compleja-conjugada-y-traspuesta-conjugada/>).

Entrada: La entrada consiste en un número entero  $n$  que será el orden de la matriz cuadrada, luego  $n^2$  entradas que serán las coordenadas de dicha matriz.

Salida: La salida será un string afirmando si la matriz es o no simétrica o hermitiana, según corresponda.

Ejemplo de entrada: 3

$$\begin{pmatrix} 5 & 1 & 3 \\ 1 & 8 & 2 \\ 3 & 2 & 5 \end{pmatrix} \quad (12)$$

Ejemplo de salida: Sí, es simétrica

Ejemplo de entrada: 2

$$\begin{pmatrix} 5 & 3+7j \\ 3-7j & 2 \end{pmatrix} \quad (13)$$

Ejemplo de salida: Sí, es hermitiana

---

**Solución:**

```
1.      # Problema 1
import numpy as np

def creaVector(n):
    A=np.zeros(n)
    for i in range(n):
        A[i]=float(input(f'A[{i}]='))
    return A

V=creaVector(12)
for i in range(-1,-len(V)-1,-1):
    print(V[i], end=' ')

2.      # Problema 2
import numpy as np

def creaVector(n):
    A=np.zeros(n)
    for i in range(n):
        A[i]=float(input(f'A[{i}]='))
    return A

def suma(X):
    s=0
    for e in X:
        s=s+e
    return s

C=creaVector(16)
print(suma(C[:8]), suma(C[8:]))
```

---

```

3.      # PROBLEMA 3
      import numpy as np

      def leeValida(m,M,msg):
          v=int(input(msg))
          while v<m or v>M:
              v=int(input(f'Error, rango permitido entre {m} y {M}, {msg}'))
          return v

      def creaMatriz(n,m,nM):
          V=np.zeros((n,m),dtype=int)
          for i in range(n):
              for j in range(m):
                  V[i,j]=int(input(f'{nM} [{i},{j}]='))
          return V

      def sumaMatrices(A,B):
          C=np.zeros((len(A),len(A[0])),dtype=int)
          for i in range(len(A)):
              for j in range(len(A[0])):
                  C[i,j]=A[i,j]+B[i,j]
          return C

      n=leeValida(2,float('inf'),'n=')
      A=creaMatriz(n,n,'A')
      B=creaMatriz(n,n,'B')
      C=sumaMatrices(A,B)
      print(C)

```

```

4.      # PROBLEMA 4
      import numpy as np

      def leeValida(m,M,msg):
          v=int(input(msg))
          while v<m or v>M:
              v=int(input(f'Error, rango permitido entre {m} y {M}, {msg}'))
          return v

      def creaMatriz(n,m,nM):
          V=np.zeros((n,m),dtype=int)
          for i in range(n):
              for j in range(m):
                  V[i,j]=int(input(f'{nM} [{i},{j}]='))
          return V

      def multiplicaMatrices(A,B):
          n,p,m=len(A),len(A[0]),len(B)
          C=np.zeros((n,m),dtype=int)
          for i in range(n):
              for j in range(m):

```

---

```

        for k in range(p):
            C[i,j]+=A[i,k]*B[k,j]
    return C

n=leeValida(2,float('inf'),'n=')
A=creaMatriz(n,m,'A')
B=creaMatriz(m,n,'B')
C=multiplicaMatrices(A,B)
print(C)

```

5.

# PROBLEMA 7

```

import numpy as np

precio=np.zeros(5, dtype=np.int16)
for i in range(5):
    precio[i]=int(input(f'Ingrese precio de artículo {i+1}='))
ventas=np.zeros((5,4), dtype=np.int16)
for i in range(5):
    for j in range(4):
        ventas[i,j]=int(input(f'Ingrese cantidad de producto {i+1} vendida en sucursal {j+1}='))
print('Cantidad de artículos vendidos en sucursal 2=', sum(ventas[:,1]))
print('Cantidad de artículos vendidos en sucursal 3=', sum(ventas[:,2]))
print('Cantidad vendida de artículo 3 en sucursal 1=', ventas[2,0])
t4=0
for i in range(5):
    t4=t4+ventas[i,3]*precio[i]
print('Recaudación total de la sucursal 4=', t4)
for i in range(5):
    for j in range(3):
        t4=t4+ventas[i,j]*precio[i]
print('Recaudación total de la empresa=', t4)

```