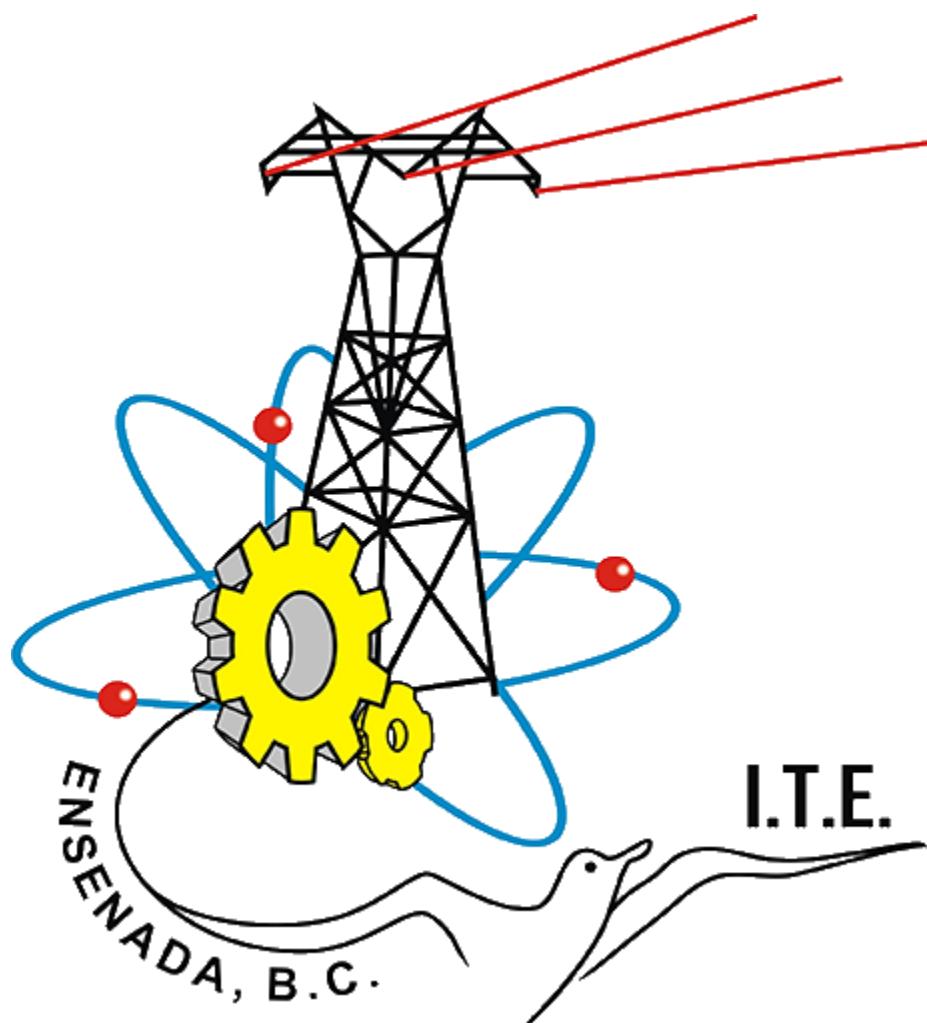


Instituto Tecnológico de Ensenada



Trabajo: Estructura de Matrices

Franyutty Gomez Angel Francisco

3SS Sistemas Computacionales

Ensenada B.C a 05 de octubre de 2025

Anexo el link de mi repositorio: <https://github.com/Franyutty9716/Estructura-de-Datos-commits?author=Franyutty9716>

### Reporte:

#### 1. Manejo de Dimensiones y Límites de Matrices

Problema: En los ejercicios iniciales (1-4), el manejo incorrecto de índices causaba accesos fuera de los límites de la matriz, resultando en comportamientos indefinidos y corrupción de memoria.

Solución: Se implementó el uso de constantes simbólicas (#define FILAS 4, #define COLUMNAS 4) para estandarizar dimensiones y evitar errores de "off-by-one". Además, se añadió validación de índices en funciones críticas.

// Antes: Problema con índices

```
for(i = 0; i <= 4; i++) // Acceso fuera de límites
```

// Después: Solución

```
for(i = 0; i < FILAS; i++) // Límites correctos
```

#### 2. Gestión de Memoria en Operaciones entre Múltiples Matrices

Problema: En los ejercicios 7-9 (operaciones entre matrices), el manejo simultáneo de tres matrices (A, B, C) complicaba el seguimiento de la memoria y las dimensiones.

Solución: Se implementó un sistema de verificación de compatibilidad dimensional antes de realizar operaciones, especialmente crítico en la multiplicación de matrices (Ejercicio 9).

```
int verificarMultiplicacion(int colA, int filB) {  
    return colA == filB; // Verificación preventiva  
}
```

#### 3. Conversión de Tipos y Precisión Numérica

Problema: En el Ejercicio 4 (cálculo de promedio), la división entre enteros producía truncamiento, dando resultados incorrectos.

Solución: Se implementó conversión explícita a float antes de la división:

```
// Solución al problema de precisión
```

```
float promedio = (float)suma / (FILAS * COLUMNAS); // ✓ Conversión correcta
```

#### 4. Validación Robusta de Entrada del Usuario

Problema: En los Ejercicios 2 y 12, la entrada de datos del usuario mediante scanf() era vulnerable a entradas incorrectas que podían bloquear el programa.

Solución: Se desarrolló una función de validación robusta con manejo de buffer y límite de intentos:

```
int leerEnteroValidado(const char* mensaje) {  
    int valor;  
    int intentos = 0;  
  
    while(intentos < MAX_INTENTOS) {  
        printf("%s", mensaje);  
  
        if(scanf("%d", &valor) == 1) {  
            return valor;  
        } else {  
            while(getchar() != '\n'); // Limpiar buffer  
            intentos++;  
        }  
    }  
}
```

```
    return 0; // Valor por defecto tras múltiples errores  
}
```

## 5. Implementación de Multiplicación de Matrices

Problema: El Ejercicio 9 requería tres bucles anidados con índices complejos, siendo propenso a errores en el orden de los índices.

Solución: Se implementó con nombres de variables descriptivos y verificación paso a paso:

```
for(int i = 0; i < FILAS_A; i++) {  
  
    for(int j = 0; j < COLUMNAS_B; j++) {  
  
        for(int k = 0; k < COLUMNAS_A; k++) { // k conecta las dimensiones  
  
            C[i][j] += A[i][k] * B[k][j];  
  
        }  
  
    }  
  
}
```

## 6. Cálculo de Diagonales en Matriz Cuadrada

Problema: En el Ejercicio 11, la identificación de la diagonal secundaria requería una fórmula contraintuitiva ( $i + j == n-1$ ).

Solución: Se implementó visualización del patrón y cálculo explícito:

```
// Diagonal secundaria  
  
int j = DIMENSION - 1 - i; // Fórmula para índice de columna  
  
diagonalSecundaria[i] = matriz[i][j];
```

## 7. Transposición de Matrices No Cuadradas

Problema: En el Ejercicio 10, la transposición de matriz 3x4 a 4x3 requería cambio de dimensiones, complicando el manejo de memoria.

Solución: Se utilizaron dimensiones separadas para matriz original y transpuesta:

```
#define FILAS_A 3  
  
#define COLUMNAS_A 4  
  
#define FILAS_AT_COLUMNAS_A // 4  
  
#define COLUMNAS_AT_FILAS_A // 3
```

## 8. Modularización y Reutilización de Código

Problema: Repetición de código en funciones comunes como imprimirMatriz() y llenarMatrizAleatoria() entre ejercicios.

Solución: Se estandarizaron funciones comunes y se creó un archivo de cabecera matrices.h para compartir funcionalidades entre ejercicios.

## 9. Mantenimiento de Legibilidad en Código Complejo

Problema: A medida que los ejercicios aumentaban en complejidad, el código se volvía más difícil de leer y mantener.

Solución: Se implementó:

- Comentarios estilo Doxygen para documentación
- Nombres de funciones descriptivos
- Separación clara entre lógica de presentación y cálculo
- Estructura consistente en todos los ejercicios

## 10. Gestión de Proyecto con Múltiples Archivos

Problema: Compilación manual tediosa de 12 archivos individuales.

Solución: Se desarrolló script de compilación automática (compilar.sh) que:

- Compila todos los ejercicios

- Gestiona dependencias
- Proporciona feedback claro del proceso

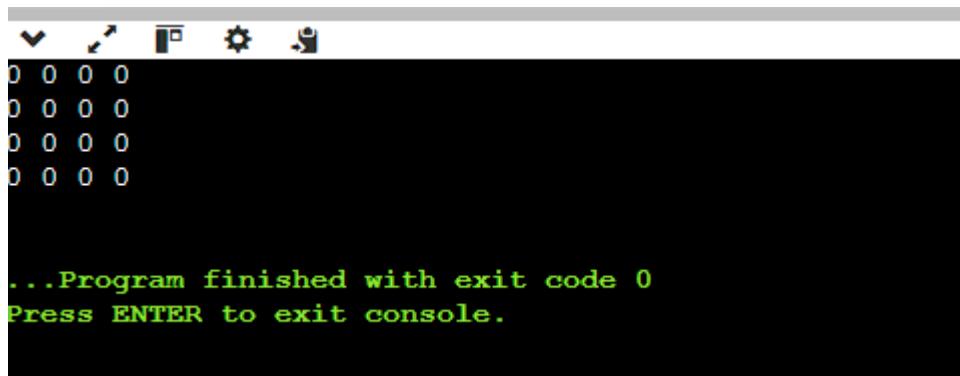
## Conclusión

Los desafíos enfrentados durante el desarrollo de estos ejercicios transformaron la comprensión teórica de matrices en habilidades prácticas de implementación. Los mayores aprendizajes surgieron de la necesidad de balancear eficiencia algorítmica con robustez del código, y de la importancia de la planificación anticipada en operaciones complejas como la multiplicación de matrices.

El proyecto demostró que el dominio de matrices en C va más allá de la sintaxis, requiriendo comprensión profunda de gestión de memoria, diseño de algoritmos y principios de ingeniería de software. Cada ejercicio superado representó no solo un concepto aprendido, sino una mejor práctica incorporada al toolkit de desarrollo.

Capturas de Pantalla:

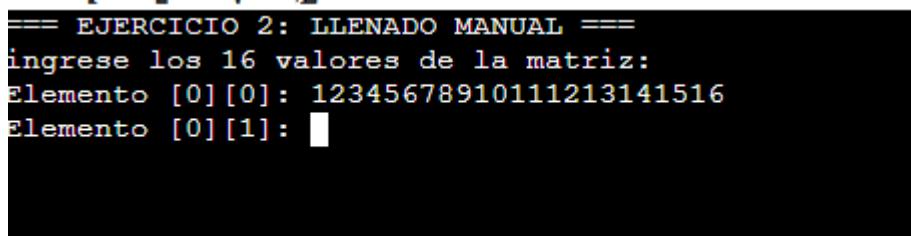
Ejercicio 1:



```
0 0 0 0
0 0 0 0
0 0 0 0
0 0 0 0

...Program finished with exit code 0
Press ENTER to exit console.
```

Ejercicio 2:



```
== EJERCICIO 2: LLENADO MANUAL ==
ingrese los 16 valores de la matriz:
Elemento [0][0]: 12345678910111213141516
Elemento [0][1]: 
```

Ejercicio 3:

```
==== EJERCICIO 3: MATRIZ ALEATORIA ====
Generando matriz 4x4 con numeros aleatorios (0-9)...

Matriz generada:
7      3      7      9
6      2      0      9
2      0      4      0
4      0      5      4

==== ESTADÍSTICAS ====
Números pares: 10
Números impares: 6
Total de elementos: 16

¡Ejercicio 3 completado!

...Program finished with exit code 0
Press ENTER to exit console.
```

```
==== EJERCICIO 4: SUMA Y PROMEDIO ====
Generando matriz 4x4 con números aleatorios (0-9)...

Matriz generada:
2      2      7      0
8      6      3      3
1      1      3      9
9      4      6      4

==== RESULTADOS ====
Suma de todos los elementos: 68
Promedio de los elementos: 4.25
Total de elementos: 16

Cálculo: 68 / 16 = 4.25

¡Ejercicio 4 completado!

...Program finished with exit code 0
Press ENTER to exit console.
```

```
==== EJERCICIO 5: MÁXIMO Y MÍNIMO ====
Generando matriz 4x4 con números aleatorios (0-99)...

Matriz generada:
11      56      54      45
29      93      63      52
98      34      70      89
54      33      16      82

==== ELEMENTOS EXTREMOS ====
Máximo: 98 en posición [2][0]
Mínimo: 11 en posición [0][0]

Matriz con elementos destacados:
[min:11]      56      54      45
 29      93      63      52
[MÁX:98]      34      70      89
 54      33      16      82

==== ESTADÍSTICAS COMPLETAS ====
Rango de valores: 11 - 98
Amplitud: 87
Suma total: 879

¡Ejercicio 5 completado!

...Program finished with exit code 0
Press ENTER to exit console.
```

```
==== EJERCICIO 6: SUMA POR FILAS Y COLUMNAS ====  
  
==== EJEMPLO ESPERADO ====  
Matriz:  
5  8  3  2  | Suma fila 0: 18  
1  9  4  7  | Suma fila 1: 21  
6  2  5  3  | Suma fila 2: 16  
4  7  1  8  | Suma fila 3: 20  
  
Suma columnas: 16  26  13  20  
  
--- GENERANDO MATRIZ ALEATORIA ---  
  
Matriz:  
-----  
| 5  4  5  2  | → Suma fila 0: 16  
| 4  2  7  9  | → Suma fila 1: 22  
| 9  5  0  0  | → Suma fila 2: 14  
| 6  8  6  0  | → Suma fila 3: 20  
  
Suma columnas: 24  19  18  11  
  
¡Ejercicio 6 completado!  
  
...Program finished with exit code 0  
Press ENTER to exit console.
```

```
== EJERCICIO 7: SUMA DE MATRICES ==
C[i][j] = A[i][j] + B[i][j]
```

Matriz A:

```
| 7  3  8  7 |
| 5  0  9  0 |
| 7  4  8  8 |
| 4  2  5  0 |
```

Matriz B:

```
| 7  2  3  1 |
| 9  7  0  5 |
| 6  6  2  1 |
| 6  5  4  5 |
```

Matriz C:

```
| 14  5  11  8 |
| 14  7  9  5 |
| 13  10  10  9 |
| 10  7  9  5 |
```

Fórmula matemática:

```
C[i][j] = A[i][j] + B[i][j]
```

```
Donde i = 0..3, j = 0..3
```

```
== DEMOSTRACIÓN DE CÁLCULO ==
```

Ejemplo para algunos elementos:

```
C[0][0] = A[0][0] + B[0][0] = 7 + 7 = 14
```

```
C[1][1] = A[1][1] + B[1][1] = 0 + 7 = 7
```

```
C[2][2] = A[2][2] + B[2][2] = 8 + 2 = 10
```

```
C[3][3] = A[3][3] + B[3][3] = 0 + 5 = 5
```

;Ejercicio 7 completado!

```
...Program finished with exit code 0
```

```
Press ENTER to exit console.
```

```

==== EJERCICIO 8: RESTA DE MATRICES ====
C[i][j] = A[i][j] - B[i][j]

Matriz A:
| 4   1   4   0   |
| 0   8   8   1   |
| 8   5   6   5   |
| 7   1   5   7   |

Matriz B:
| 7   3   5   7   |
| 7   6   5   6   |
| 3   7   4   8   |
| 3   3   1   7   |

==== PROCESO DE RESTA ====
| 4   1   4   0   |      | 7   3   5   7   |      | -3   -2   -1   -7   |
| 0   8   8   1   | - | 7   6   5   6   | = | -7   2   3   -5   |
| 8   5   6   5   |      | 3   7   4   8   |      | 5   -2   2   -3   |
| 7   1   5   7   |      | 3   3   1   7   |      | 4   -2   4   0   |

Matriz C:
| -3   -2   -1   -7   |
| -7   2   3   -5   |
| 5   -2   2   -3   |
| 4   -2   4   0   |

==== DEMOSTRACIÓN DE CÁLCULO ====
Fórmula: C[i][j] = A[i][j] - B[i][j]

C[0][0] = A[0][0] - B[0][0] = 4 - 7 = -3
C[1][1] = A[1][1] - B[1][1] = 8 - 6 = 2
C[2][2] = A[2][2] - B[2][2] = 6 - 4 = 2
C[3][3] = A[3][3] - B[3][3] = 7 - 7 = 0

¡Ejercicio 8 completado!

...Program finished with exit code 0
Press ENTER to exit console.

```

```
==== EJERCICIO 9: MULTIPLICACIÓN DE MATRICES ===
```

```
==== EJEMPLO DEL ENUNCIADO ===
```

A (2x3)	B (3x2)	C (2x2)
1 2 3	1 4	22 28
4 5 6	2 5	49 64
	3 6	

Las matrices se pueden multiplicar:

$$A(2 \times 3) \times B(3 \times 2) = C(2 \times 2)$$

Matriz A (2x3) :

	1	2	3	
	4	5	6	

Matriz B (3x2) :

	1	4	
	2	5	
	3	6	

Matriz C (2x2) :

	14	32	
	32	77	

```
==== REGLAS DE MULTIPLICACIÓN ===
```

Para multiplicar  $A(m \times n) \times B(n \times p)$ :

- Las columnas de A deben igualar las filas de B
- El resultado C tendrá dimensiones  $(m \times p)$
- Fórmula:  $C[i][j] = \sum(A[i][k] \times B[k][j])$  para  $k = 0..n-1$

```
==== PROCESO DE CÁLCULO ===
```

$$C[0][0] = (1 \times 1) + (2 \times 2) + (3 \times 3) = 14$$

$$C[0][1] = (1 \times 4) + (2 \times 5) + (3 \times 6) = 32$$

$$C[1][0] = (4 \times 1) + (5 \times 2) + (6 \times 3) = 32$$

$$C[1][1] = (4 \times 4) + (5 \times 5) + (6 \times 6) = 77$$

¡Ejercicio 9 completado!

```
...Program finished with exit code 0
```

```
Press ENTER to exit console.
```

```
==== EJERCICIO 10: MATRIZ TRANSPUESTA ====
Generando matriz A de 3x4 con valores aleatorios...

Matriz A (3x4):
| 2  8  9  8 |
| 0  8  7  4 |
| 5  9  6  2 |

Matriz AT (Transpuesta) (4x3):
| 2  0  5 |
| 8  8  9 |
| 9  7  6 |
| 8  4  2 |

==== COMPARACIÓN DE DIMENSIONES ====
Matriz original A: 3x4
Matriz transpuesta AT: 4x3
Se intercambian filas por columnas.

==== PROCESO DE TRANSPOSICIÓN ====
Fórmula: AT[j][i] = A[i][j]

Intercambios realizados:
AT[0][0] = A[0][0] = 2
AT[1][0] = A[0][1] = 8
AT[2][0] = A[0][2] = 9
AT[3][0] = A[0][3] = 8
AT[0][1] = A[1][0] = 0
AT[1][1] = A[1][1] = 8
AT[2][1] = A[1][2] = 7
AT[3][1] = A[1][3] = 4
AT[0][2] = A[2][0] = 5
AT[1][2] = A[2][1] = 9
AT[2][2] = A[2][2] = 6
AT[3][2] = A[2][3] = 2

¡Ejercicio 10 completado!

...Program finished with exit code 0
Press ENTER to exit console.[]
```

```

==== EJERCICIO 11: DIAGONAL PRINCIPAL Y SECUNDARIA ====
Generando matriz cuadrada 4x4 con valores aleatorios...

Matriz 4x4:
| [P:6] 7   8   [S:7] |
| 6   [P:4] [S:7] 1   |
| 2   [S:8] [P:1] 4   |
| [S:3] 1   3   [P:9] |

==== DIAGONAL PRINCIPAL ====
Elementos (donde i == j): A[0][0]=6, A[1][1]=4, A[2][2]=1, A[3][3]=9
Suma: 6 + 4 + 1 + 9 = 20

==== DIAGONAL SECUNDARIA ====
Elementos (donde i + j == 3): A[0][3]=7, A[1][2]=7, A[2][1]=8, A[3][0]=3
Suma: 7 + 7 + 8 + 3 = 25

==== COMPARACIÓN ====
Suma diagonal principal: 20
Suma diagonal secundaria: 25
La diagonal principal tiene mayor suma.

==== POSICIONES DE LAS DIAGONALES ====
Matriz 4x4:
| P . . S |
| . P S . |
| . S P . |
| S . . P |
Leyenda: P=Principal, S=Secundaria, X=Ambas

¡Ejercicio 11 completado!

```

...Program finished with exit code 0  
Press ENTER to exit console.

```

==== EJERCICIO 12: BUSCAR ELEMENTO EN MATRIZ ====
Generando matriz 4x4 con valores aleatorios...

```

```

Matriz generada:
    Col0 Col1 Col2 Col3
    -----
F0 |  5   0   0   3 |
F1 |  5   8   4   2 |
F2 |  8   1   2   8 |
F3 |  6   0   1   8 |

```

Ingrese el número a buscar: