**Universidad de las Fuerzas Armadas ESPE**

**Departamento: Ciencias de la computación**

**Carrera: Ingeniería en Electricidad y Automatización**

**Taller académico N°:3 U2**

---

## 1. Información General

- **Asignatura: Fundamentos de Programación**
- **Apellidos y nombres de los estudiantes: Ana Ulloa, Jordy Cevallos, Bryan Miguitama.**
- **NRC: 20823**
- **Fecha de realización: 10/06/2025**

---

## 2. Objetivo del Taller y Desarrollo

### Objetivo del Taller:

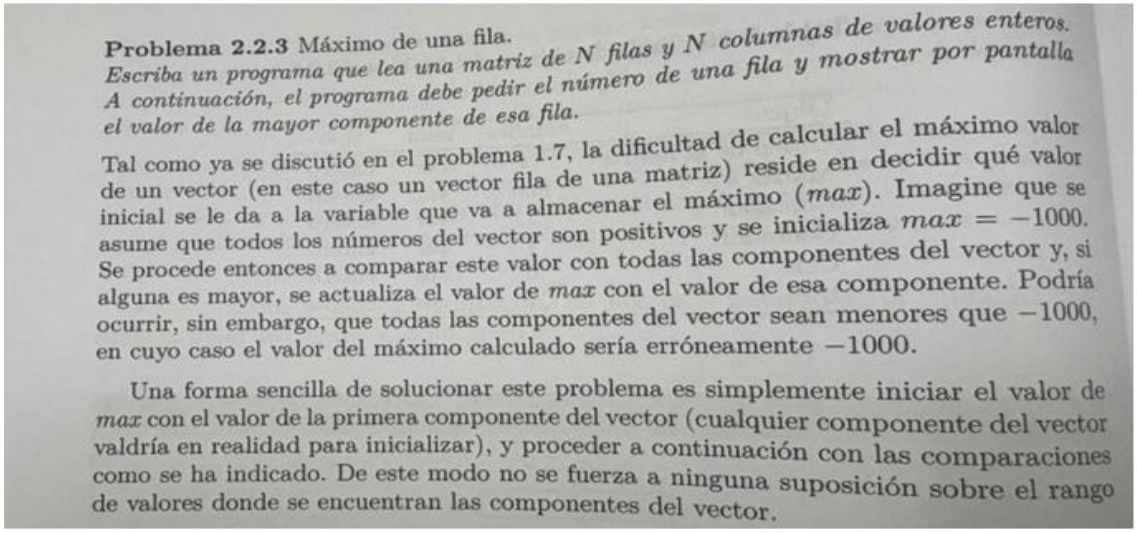
Comprender y aplicar los conceptos fundamentales del manejo de matrices en programación, mediante el desarrollo de ejercicios prácticos orientados a identificar, manipular y analizar datos en estructuras bidimensionales.

### Desarrollo:

En este taller se trabajó con matrices cuadradas, ingresando datos desde teclado y utilizando estructuras repetitivas para recorrer filas y columnas. Se implementaron algoritmos en pseudocódigo para encontrar el valor máximo de una fila específica, validando los índices y corrigiendo errores comunes relacionados con el rango de subíndices en PSeInt.

---

**Taller N° 3 U2**



**Problema 2.2.3** Máximo de una fila.
*Escriba un programa que lea una matriz de N filas y N columnas de valores enteros. A continuación, el programa debe pedir el número de una fila y mostrar por pantalla el valor de la mayor componente de esa fila.*

Tal como ya se discutió en el problema 1.7, la dificultad de calcular el máximo valor de un vector (en este caso un vector fila de una matriz) reside en decidir qué valor inicial se le da a la variable que va a almacenar el máximo ($max$). Imagine que se asume que todos los números del vector son positivos y se inicializa $max = -1000$. Se procede entonces a comparar este valor con todas las componentes del vector y, si alguna es mayor, se actualiza el valor de $max$ con el valor de esa componente. Podría ocurrir, sin embargo, que todas las componentes del vector sean menores que $-1000$, en cuyo caso el valor del máximo calculado sería erróneamente $-1000$.

Una forma sencilla de solucionar este problema es simplemente iniciar el valor de $max$ con el valor de la primera componente del vector (cualquier componente del vector valdría en realidad para inicializar), y proceder a continuación con las comparaciones como se ha indicado. De este modo no se fuerza a ninguna suposición sobre el rango de valores donde se encuentran las componentes del vector.

1. Requisitos Funcionales

- Permitir al usuario ingresar un número entero N*N que representa el tamaño de la matriz.
- Debe solicitar al usuario el ingreso de N elementos para llenar la matriz.
- Debe mostrar la posición actual del elemento a ingresar (fila, columna) para guiar al usuario.
- El sistema debe recorrer cada fila de la matriz y comparar todos los elementos de esa fila para identificar el valor máximo.
- Para cada fila, el sistema debe almacenar temporalmente el valor máximo encontrado.
- Debe mostrar en pantalla el valor máximo correspondiente a cada fila, indicando el número de fila al que pertenece.

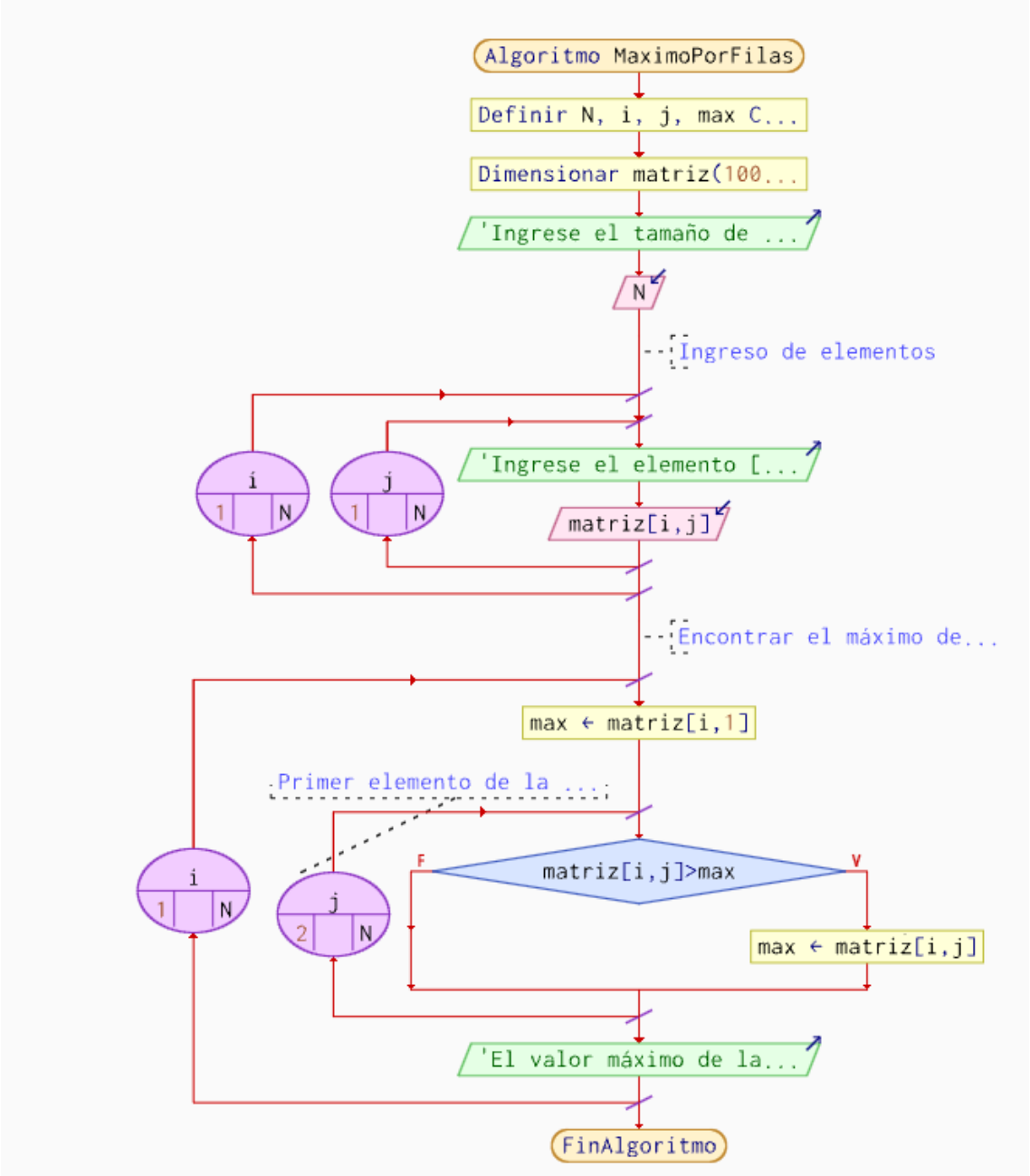
2. TABLA DE OBJETOS

| objeto | Nombre | Tipo | Valor |
|---|---|---|---|
| Dato 1 | N | Entero | variable |
| Dato 2 | i | Entero | variables |
| Dato 3 | j | Entero | variable |
| Dato 4 | max | Entero | variable |
| Dato 5 | matriz | Entero | variable |

3. PSEUDOCODIGO

```
1. Algoritmo MaximoPorFilas
2. Definir N, i, j, max Como Entero
3. Dimension matriz[100, 100]
4. Escribir "Ingrese el tamaño de la matriz (N x N):"
5. Leer N
6. // Ingreso de elementos
7. Para i <- 1 Hasta N Hacer
8. Para j <- 1 Hasta N Hacer
9. Escribir "Ingrese el elemento [", i, ",", j, "]:"
10. Leer matriz[i, j]
11. FinPara
12. FinPara
13. // Encontrar el máximo de cada fila
14. Para i <- 1 Hasta N Hacer
15. max <- matriz[i, 1] // Primer elemento de la fila
16. Para j <- 2 Hasta N Hacer
17. Si matriz[i, j] > max Entonces
18. max <- matriz[i, j]
19. FinSi
20. FinPara
21. Escribir "El valor máximo de la fila ", i, " es: ", max
22. FinPara
23. FinAlgoritmo
```

4. Diagrama de flujo



5. Prueba de Escritorio

| | Columna 1 | Columna 2 | Columna 3 |
|---|---|---|---|
| Fila 1 | 5 | 2 | 9 |
| Fila 2 | 4 | 8 | 1 |
| Fila 3 | 7 | 6 | 3 |

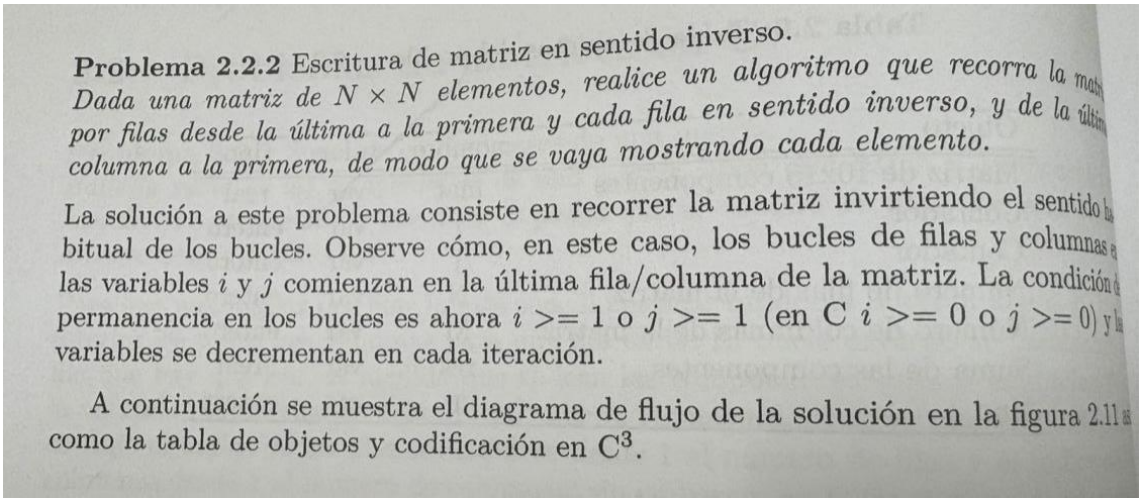
| Fila | Valores de `matriz[i, j]` | `max` | Procedimiento | Resultado |
|---|---|---|---|---|
| 1 | 5, 2, 9 | 5 | 2 < 5 → no cambia9 > 5 → max = 9 | El valor máximo de la fila 1 es: **9** |
| 2 | 4, 8, 1 | 4 | 8 > 4 → max = 81 < 8 → no cambia | El valor máximo de la fila 2 es: **8** |
| 3 | 7, 6, 3 | 7 | 6 < 7 → no cambia3 < 7 → no cambia | El valor máximo de la fila 3 es: **7** |

---

6. Codeblok

```
#include <stdio.h>

int main() {
    int N, i, j, max;
    int matriz[100][100];

    // Ingreso del tamaño de la matriz
    printf("Ingrese el tamano de la matriz (N x N): ");
    scanf("%d", &N);

    // Ingreso de los elementos de la matriz
    for (i = 0; i < N; i++) {
        for (j = 0; j < N; j++) {
            printf("Ingrese el elemento [%d, %d]: ", i + 1, j + 1);
            scanf("%d", &matriz[i][j]);
        }
    }

    // Encontrar el máximo de cada fila
    for (i = 0; i < N; i++) {
        max = matriz[i][0]; // Primer elemento de la fila
        for (j = 1; j < N; j++) {
            if (matriz[i][j] > max) {
                max = matriz[i][j];
            }
        }
```

```
        printf("El valor maximo de la fila %d es: %d\n", i + 1, max);
    }

    return 0;
}
```



**Problema 2.2.2** Escritura de matriz en sentido inverso.
*Dada una matriz de $N \times N$ elementos, realice un algoritmo que recorra la matriz por filas desde la última a la primera y cada fila en sentido inverso, y de la última columna a la primera, de modo que se vaya mostrando cada elemento.*

La solución a este problema consiste en recorrer la matriz invirtiendo el sentido habitual de los bucles. Observe cómo, en este caso, los bucles de filas y columnas en las variables $i$ y $j$ comienzan en la última fila/columna de la matriz. La condición de permanencia en los bucles es ahora $i >= 1$ o $j >= 1$ (en C $i >= 0$ o $j >= 0$) y las variables se decrementan en cada iteración.

A continuación se muestra el diagrama de flujo de la solución en la figura 2.11 así como la tabla de objetos y codificación en C³.

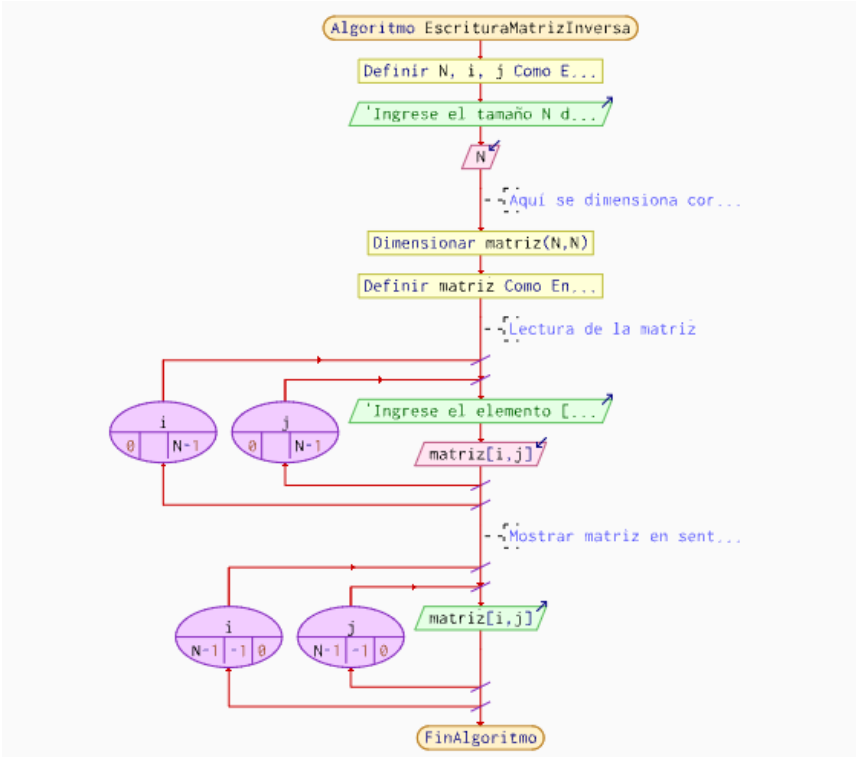
- Requisito funcional:
  - “El sistema debe permitir al usuario ingresar una matriz cuadrada de tamaño N×N y mostrar sus elementos en sentido inverso, es decir, desde la última fila y última columna hasta la primera, respetando el orden inverso de recorrido”.
- Tabla de objetos:

| Objeto | Nombre | Tipo | Valor |
|---|---|---|---|
| D1 | I | entero | Variable |
| D2 | J | entero | Variable |
| D3 | N | entero | Variable |
| D4 | Matriz | entero | Variable |
| D5 | Resultado | entero | Variable |

- Pseint :

```
Algoritmo EscrituraMatrizInversa
    Definir N, i, j como Entero

    Escribir "Ingrese el tamaño N de la matriz:"
    Leer N

    // Aquí se dimensiona correctamente la matriz con N×N
    Dimension matriz[N, N]
    Definir matriz como Entero

    // Lectura de la matriz
    Para i ← 0 Hasta N - 1
        Para j ← 0 Hasta N - 1
            Escribir "Ingrese el elemento [", i, ",", j, "]:"
            Leer matriz[i, j]
        FinPara
    FinPara

    // Mostrar matriz en sentido inverso
    Para i ← N - 1 Hasta 0 Con Paso -1
        Para j ← N - 1 Hasta 0 Con Paso -1
            Escribir matriz[i, j]
        FinPara
    FinPara
FinProceso
```

- Diagrama de flujo :



- Código en C codeblocks:

```
#include <stdio.h>

#define MAX 100  // Tamaño máximo permitido para la matriz

int main() {
   int N, i, j;
   int matriz[MAX][MAX];  // Declaramos la matriz con tamaño fijo

   // Pedir el tamaño de la matriz
   printf("Ingrese el tamaño N de la matriz (máximo %d): ", MAX);
   scanf("%d", &N);

   // Validar que N no sobrepase el tamaño máximo
   if (N > MAX || N <= 0) {
      printf("Error: tamaño inválido.\n");
      return 1;
   }

   // Leer los elementos de la matriz
   for (i = 0; i < N; i++) {
      for (j = 0; j < N; j++) {
         printf("Ingrese el elemento [%d,%d]: ", i, j);
         scanf("%d", &matriz[i][j]);
      }
   }

   // Mostrar la matriz en orden inverso (de abajo a arriba, derecha a izquierda)
   printf("\nMatriz en sentido inverso:\n");
   for (i = N - 1; i >= 0; i--) {
      for (j = N - 1; j >= 0; j--) {
         printf("%d ", matriz[i][j]);
      }
      printf("\n");
   }

   return 0;
}
```