



Carátula para entrega de prácticas

Facultad de Ingeniería

Laboratorios de docencia

Laboratorio de Computación Salas A y B

Profesor(a): **MANUEL ENRIQUE CASTAÑEDA CASTAÑEDA**

Asignatura: **FUNDAMENTOS DE PROGRAMACION**

Grupo: **18**

No de Práctica(s): **PRACTICA 10**

Integrante(s): **MORENO LOERA DIEGO**

No. de lista o brigada: **32**

Semestre: **SEMESTRE 2025-1**

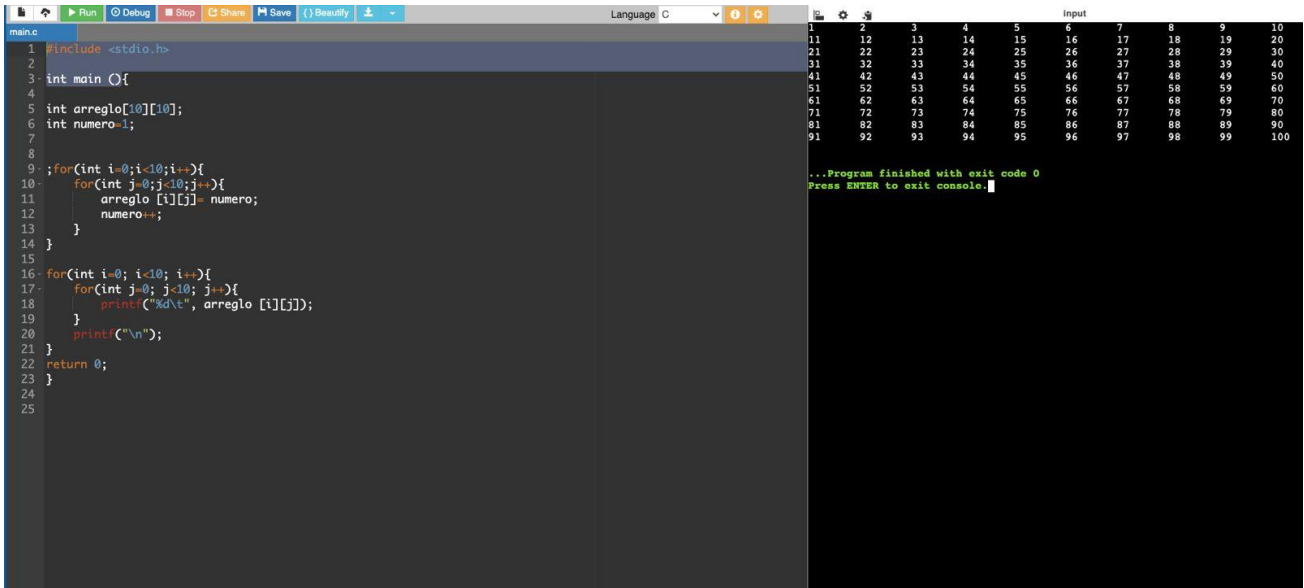
Fecha de entrega: **28 DE OCTUBRE DEL 2024**

Observaciones:

CALIFICACIÓN: _____

DESARROLLO DE LA PRACTICA 10

1. Que muestre los primeros 100 números de izquierda a derecha usando un arreglo de dos dimensiones.

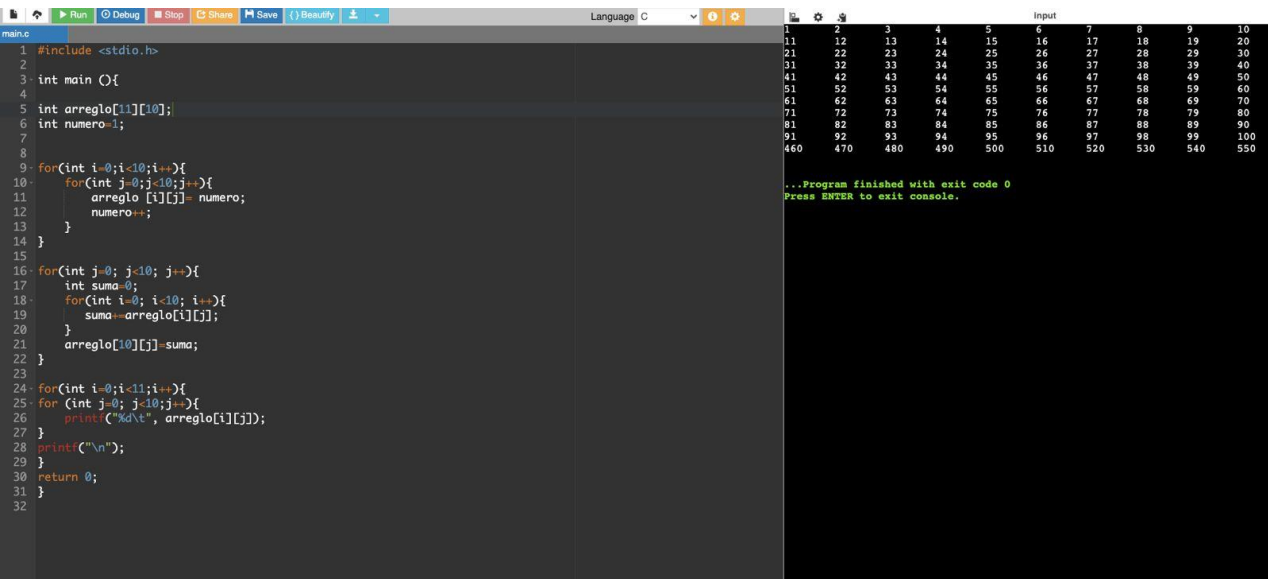


```
1 #include <stdio.h>
2
3 int main (){
4
5     int arreglo[10][10];
6     int numero=1;
7
8
9     for(int i=0; i<10; i++){
10         for(int j=0; j<10; j++){
11             arreglo[i][j]= numero;
12             numero++;
13         }
14     }
15
16     for(int i=0; i<10; i++){
17         for(int j=0; j<10; j++){
18             printf("%d\t", arreglo[i][j]);
19         }
20         printf("\n");
21     }
22     return 0;
23 }
24
25
```

1	2	3	4	5	6	7	8	9	10
11	12	13	14	15	16	17	18	19	20
21	22	23	24	25	26	27	28	29	30
31	32	33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48	49	50
51	52	53	54	55	56	57	58	59	60
61	62	63	64	65	66	67	68	69	70
71	72	73	74	75	76	77	78	79	80
81	82	83	84	85	86	87	88	89	90
91	92	93	94	95	96	97	98	99	100

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

2. Que muestre los primeros 100 números de izquierda a derecha usando un arreglo de dos dimensiones, la última fila a mostrará la suma de sus respectivas columnas.



```
1 #include <stdio.h>
2
3 int main (){
4
5     int arreglo[11][10];
6     int numero=1;
7
8
9     for(int i=0; i<10; i++){
10         for(int j=0; j<10; j++){
11             arreglo[i][j]= numero;
12             numero++;
13         }
14     }
15
16     for(int j=0; j<10; j++){
17         int suma=0;
18         for(int i=0; i<10; i++){
19             suma+=arreglo[i][j];
20         }
21         arreglo[10][j]=suma;
22     }
23
24     for(int i=0; i<11; i++){
25         for(int j=0; j<10; j++){
26             printf("%d\t", arreglo[i][j]);
27         }
28         printf("\n");
29     }
30     return 0;
31 }
32
```

1	2	3	4	5	6	7	8	9	10
11	12	13	14	15	16	17	18	19	20
21	22	23	24	25	26	27	28	29	30
31	32	33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48	49	50
51	52	53	54	55	56	57	58	59	60
61	62	63	64	65	66	67	68	69	70
71	72	73	74	75	76	77	78	79	80
81	82	83	84	85	86	87	88	89	90
91	92	93	94	95	96	97	98	99	100
460	470	480	490	500	510	520	530	540	550

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

3. Que rellene un arreglo de dos dimensiones con números pares, lo imprima en pantalla y después que pida una posición X,Y y mostrar el número correspondiente.

```
1 #include <stdio.h>
2
3 int main (){
4     int arreglo[5][5];
5     int numeropar=2;
6
7     for (int i=0; i<5; i++){
8         for (int j= 0; j <5; j++){
9             arreglo [i][j]= numeropar;
10            numeropar+=2;
11        }
12    }
13
14    printf("Arreglo de numeros pares: \n");
15    for(int i =0; i<5; i++){
16        for (int j=0; j<5;j++){
17            printf (" %d\t", arreglo[i][j]);
18        }
19        printf ("\n");
20    }
21
22    int x, y;
23    printf("Introduce la posicion x (0-4): ");
24    scanf("%d", &x);
25    printf("Introduce la posicion y (0-4): ");
26    scanf("%d", &y);
27
28    if (x >= 0 && x < 5 && y >= 0 && y < 5) {
29        printf (" El numero en la posicion [%d][%d] es : %d\n", x, y, arreglo [x][y]);
30    }
31    }else {
32        printf ("Posicion fuera de rango.\n");
33    }
34    return 0;
35 }
```

Arreglo de numeros pares:

2	4	6	8	10
12	14	16	18	20
22	24	26	28	30
32	34	36	38	40
42	44	46	48	50

Introduce la posicion x (0-4): 2
Introduce la posicion y (0-4): 3
El numero en la posicion [2][3] es : 28

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

4. Que rellene una matriz de 3x3 y muestre su traspuesta (la traspuesta se consigue intercambiando filas por columnas y viceversa).

```
1 #include <stdio.h>
2
3 #define SIZE 3
4
5 int main() {
6     int matriz[SIZE][SIZE];
7     int traspuesta[SIZE][SIZE];
8
9
10    printf("Introduce los elementos de la matriz 3x3:\n");
11    for (int i = 0; i < SIZE; i++) {
12        for (int j = 0; j < SIZE; j++) {
13            printf("Elemento [%d][%d]: ", i, j);
14            scanf("%d", &matriz[i][j]);
15        }
16    }
17
18    for (int i = 0; i < SIZE; i++) {
19        for (int j = 0; j < SIZE; j++) {
20            traspuesta[j][i] = matriz[i][j];
21        }
22    }
23
24
25    printf("\nMatriz original:\n");
26    for (int i = 0; i < SIZE; i++) {
27        for (int j = 0; j < SIZE; j++) {
28            printf("%d ", matriz[i][j]);
29        }
30        printf("\n");
31    }
32
33
34    printf("\nMatriz traspuesta:\n");
35    for (int i = 0; i < SIZE; i++) {
36        for (int j = 0; j < SIZE; j++) {
37            printf("%d ", traspuesta[i][j]);
38        }
39        printf("\n");
40    }
41
42    return 0;
43 }
44
45
```

Introduce los elementos de la matriz 3x3:

Elemento [0][0]: 6
Elemento [0][1]: 8
Elemento [0][2]: 6
Elemento [1][0]: 4
Elemento [1][1]: 3
Elemento [1][2]: 2
Elemento [2][0]: 9
Elemento [2][1]: 8
Elemento [2][2]: 9

Matriz original:

6	8	6
4	3	2
9	8	9

Matriz traspuesta:

6	4	9
8	3	8
6	2	9

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

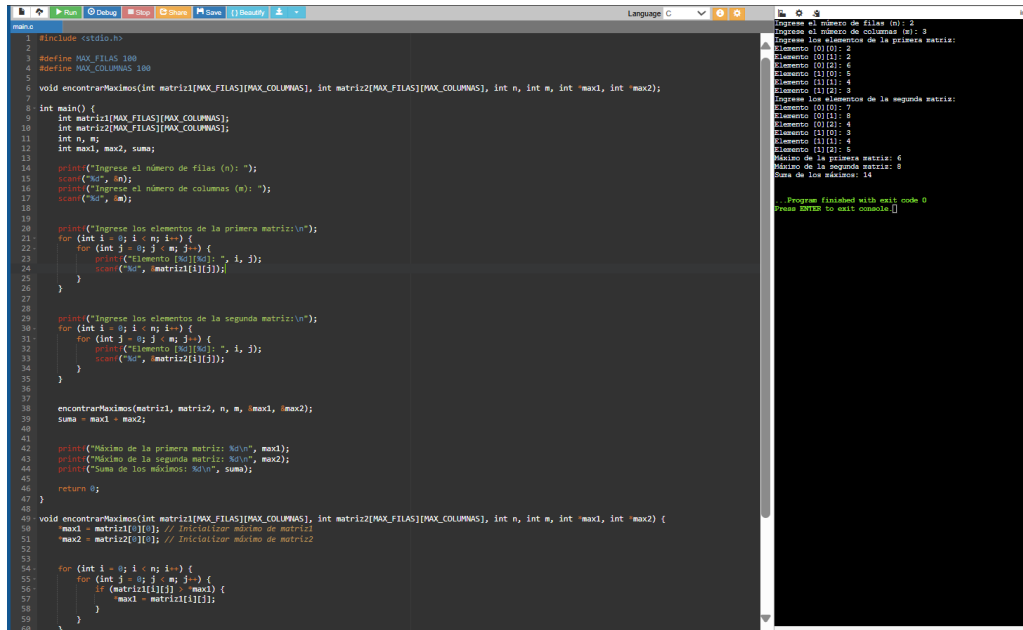
5. Que cree un arreglo de 18 X 10 indicando que poseemos una empresa de 18 vendedores cada uno de los cuales vende 10 productos. El arreglo almacena los ingresos obtenidos por cada vendedor en cada producto, de modo que un menú permite almacenar los ingresos, revisar el total de cada vendedor y obtener los ingresos totales.

```
1 // Ejercicio 5: Ingresos de vendedores y productos
2 #include <iostream>
3 using namespace std;
4
5 // Definición de constantes
6 const int VENDEDORES = 18;
7 const int PRODUCTOS = 10;
8
9 // Funciones
10 void almacenarIngresos(float ingresos[VENDEDORES][PRODUCTOS]);
11 void totalPorVendedor(float ingresos[VENDEDORES][PRODUCTOS]);
12 void totalIngresos(float ingresos[VENDEDORES][PRODUCTOS]);
13
14 int main() {
15     float ingresos[VENDEDORES][PRODUCTOS] = {}; // Inicializa el arreglo a 0
16     int opcion;
17
18     do {
19         system("clear");
20         cout << "Menu:\n";
21         cout << "1. Almacenar ingresos\n";
22         cout << "2. Revisar total de cada vendedor\n";
23         cout << "3. Obtener ingresos totales\n";
24         cout << "4. Salir\n";
25         cout << "Seleccione una opcion: ";
26         int i;
27         for (i = 1; i <= 4; i++) {
28             cout << i << " ";
29             if (i % 4 == 0) cout << "\n";
30         }
31         if (i % 4 != 0) cout << "\n";
32         int opcion;
33         while (opcion < 1 || opcion > 4) {
34             cout << "Opcion no valida, intente de nuevo.\n";
35             continue;
36         }
37         switch (opcion) {
38             case 1:
39                 almacenarIngresos(ingresos);
40                 break;
41             case 2:
42                 totalPorVendedor(ingresos);
43                 break;
44             case 3:
45                 totalIngresos(ingresos);
46                 break;
47             case 4:
48                 cout << "Saliedo del programa...\n";
49                 break;
50             default:
51                 cout << "Opcion no valida, intente de nuevo.\n";
52                 continue;
53         }
54     } while (opcion != 4);
55
56     return 0;
57 }
58
59 void almacenarIngresos(float ingresos[VENDEDORES][PRODUCTOS]) {
60     for (int i = 0; i < VENDEDORES; i++) {
61         for (int j = 0; j < PRODUCTOS; j++) {
62             cout << "Ingrese el ingreso del vendedor " << i + 1 << " para el producto " << j + 1 << ": ";
63             float ingreso;
64             while (cin.get() != '\n') continue;
65             while (cin.get() < 0) {
66                 cout << "Error: ingreso no puede ser negativo.\n";
67                 continue;
68             }
69             ingresos[i][j] = ingreso;
70         }
71     }
72 }
73
74 void totalPorVendedor(float ingresos[VENDEDORES][PRODUCTOS]) {
75     for (int i = 0; i < VENDEDORES; i++) {
76         float total = 0;
77         for (int j = 0; j < PRODUCTOS; j++) {
78             total += ingresos[i][j];
79         }
80         cout << "Total de ingresos del vendedor " << i + 1 << ": " << total << "\n";
81     }
82 }
83
84 void totalIngresos(float ingresos[VENDEDORES][PRODUCTOS]) {
85     float totalGeneral = 0;
86     for (int i = 0; i < VENDEDORES; i++) {
87         for (int j = 0; j < PRODUCTOS; j++) {
88             totalGeneral += ingresos[i][j];
89         }
90     }
91     cout << "Total de ingresos de todos los vendedores: " << totalGeneral << "\n";
92 }
```

6. Que mediante un menú admita reservar o cancelar asientos de un avión, así como mostrar qué asientos están ocupados y libreas actualmente. El arreglo tendrá 25 filas y 4 columnas.

```
1 // Ejercicio 6: Reservar y cancelar asientos de un avión
2 #include <iostream>
3 #include <string>
4 using namespace std;
5
6 // Definición de constantes
7 const int FILAS = 25;
8 const int COLUMNAS = 4;
9
10 // Funciones
11 void mostrarAsientos(char asientos[FILAS][COLUMNAS]);
12 void reservarAsiento(char asientos[FILAS][COLUMNAS], int fila, int columna);
13 void cancelarAsiento(char asientos[FILAS][COLUMNAS], int fila, int columna);
14
15 int main() {
16     char asientos[FILAS][COLUMNAS];
17     int opcion;
18
19     do {
20         system("clear");
21         cout << "Menu:\n";
22         cout << "1. Mostrar asientos\n";
23         cout << "2. Reservar asiento\n";
24         cout << "3. Cancelar asiento\n";
25         cout << "4. Salir\n";
26         cout << "Seleccione una opcion: ";
27         int i;
28         for (i = 1; i <= 4; i++) {
29             cout << i << " ";
30             if (i % 4 == 0) cout << "\n";
31         }
32         if (i % 4 != 0) cout << "\n";
33         int opcion;
34         while (opcion < 1 || opcion > 4) {
35             cout << "Opcion no valida, intente de nuevo.\n";
36             continue;
37         }
38         switch (opcion) {
39             case 1:
40                 mostrarAsientos(asientos);
41                 break;
42             case 2:
43                 reservarAsiento(asientos);
44                 break;
45             case 3:
46                 cancelarAsiento(asientos);
47                 break;
48             case 4:
49                 cout << "Saliedo del programa...\n";
50                 break;
51             default:
52                 cout << "Opcion no valida, intente de nuevo.\n";
53                 continue;
54         }
55     } while (opcion != 4);
56
57     return 0;
58 }
59
60 void mostrarAsientos(char asientos[FILAS][COLUMNAS]) {
61     for (int i = 0; i < FILAS; i++) {
62         for (int j = 0; j < COLUMNAS; j++) {
63             cout << asientos[i][j] << " ";
64             if (j % 4 == 3) cout << "\n";
65         }
66         if (j % 4 != 3) cout << "\n";
67     }
68 }
69
70 void reservarAsiento(char asientos[FILAS][COLUMNAS]) {
71     int fila, columna;
72     cout << "Ingrese fila (0-24) y columna (0-3) para reservar: ";
73     while (cin.get() != '\n') continue;
74     while (cin.get() < 0 || cin.get() > 24) {
75         cout << "Error: fila no valida.\n";
76         continue;
77     }
78     while (cin.get() < 0 || cin.get() > 3) {
79         cout << "Error: columna no valida.\n";
80         continue;
81     }
82     fila = cin.get();
83     columna = cin.get();
84     if (asientos[fila][columna] != ' ') {
85         cout << "Error: asiento ya ocupado.\n";
86         continue;
87     }
88     asientos[fila][columna] = 'X';
89     cout << "Asiento reservado exitosamente.\n";
90 }
91
92 void cancelarAsiento(char asientos[FILAS][COLUMNAS]) {
93     int fila, columna;
94     cout << "Ingrese fila (0-24) y columna (0-3) para cancelar: ";
95     while (cin.get() != '\n') continue;
96     while (cin.get() < 0 || cin.get() > 24) {
97         cout << "Error: fila no valida.\n";
98         continue;
99     }
100    while (cin.get() < 0 || cin.get() > 3) {
101        cout << "Error: columna no valida.\n";
102        continue;
103    }
104    fila = cin.get();
105    columna = cin.get();
106    if (asientos[fila][columna] != 'X') {
107        cout << "Error: asiento no reservado.\n";
108        continue;
109    }
110    asientos[fila][columna] = ' ';
111    cout << "Asiento cancelado exitosamente.\n";
112 }
```

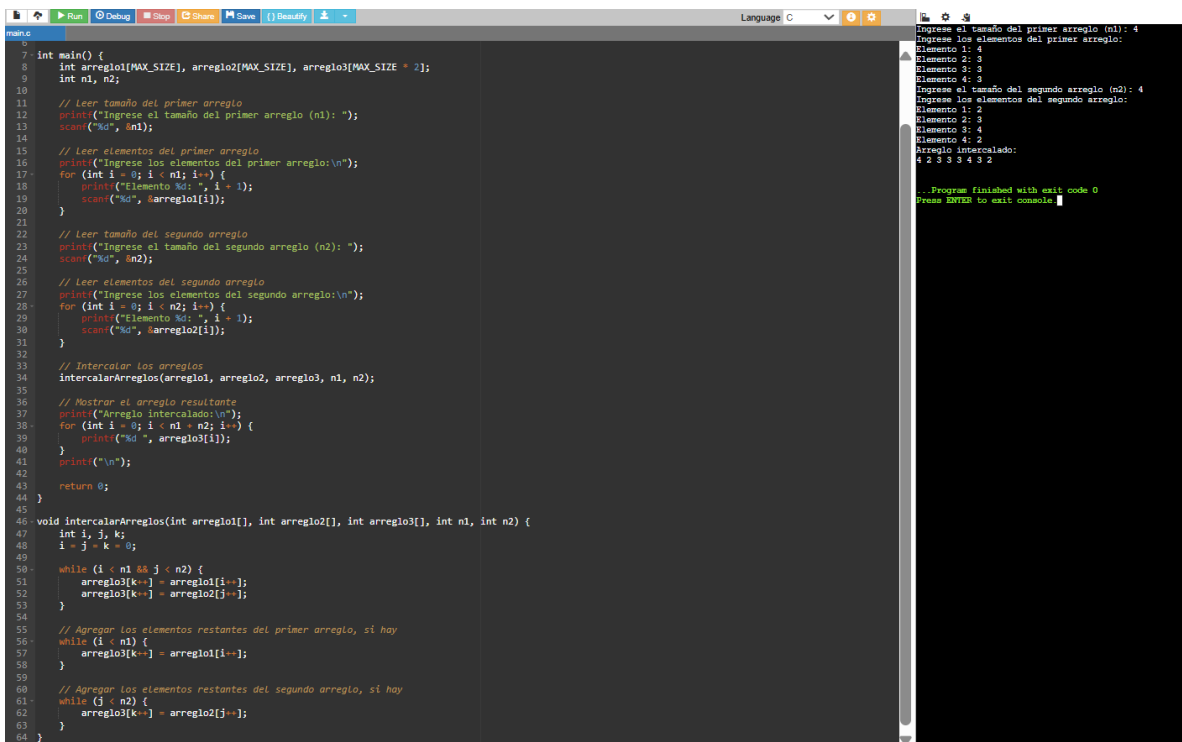
7. Que busque en dos matrices de nxm el número mayor en cada una de ellas y los sume



```
1 #include <stdio.h>
2
3 #define MAX_FILAS 100
4 #define MAX_COLUMNAS 100
5
6 void encontrarMaximos(int matriz1[MAX_FILAS][MAX_COLUMNAS], int n, int m, int *max1, int *max2);
7
8 int main() {
9     int matriz1[MAX_FILAS][MAX_COLUMNAS];
10    int matriz2[MAX_FILAS][MAX_COLUMNAS];
11    int n, m;
12    int max1, max2, suma;
13
14    printf("Ingrese el número de filas (n): ");
15    scanf("%d", &n);
16    printf("Ingrese el número de columnas (m): ");
17    scanf("%d", &m);
18
19    printf("Ingrese los elementos de la primera matriz:\n");
20    for (int i = 0; i < n; i++) {
21        for (int j = 0; j < m; j++) {
22            printf("Elemento [%d][%d]: ", i, j);
23            scanf("%d", &matriz1[i][j]);
24        }
25    }
26
27    printf("Ingrese los elementos de la segunda matriz:\n");
28    for (int i = 0; i < n; i++) {
29        for (int j = 0; j < m; j++) {
30            printf("Elemento [%d][%d]: ", i, j);
31            scanf("%d", &matriz2[i][j]);
32        }
33    }
34
35    encontrarMaximos(matriz1, matriz2, n, m, &max1, &max2);
36    suma = max1 + max2;
37
38    printf("Máximo de la primera matriz: %d\n", max1);
39    printf("Máximo de la segunda matriz: %d\n", max2);
40    printf("Suma de los máximos: %d\n", suma);
41
42    return 0;
43 }
44
45 void encontrarMaximos(int matriz1[MAX_FILAS][MAX_COLUMNAS], int matriz2[MAX_FILAS][MAX_COLUMNAS], int n, int m, int *max1, int *max2) {
46    *max1 = matriz1[0][0]; // Inicializar máximo de matriz1
47    *max2 = matriz2[0][0]; // Inicializar máximo de matriz2
48
49    for (int i = 0; i < n; i++) {
50        for (int j = 0; j < m; j++) {
51            if (matriz1[i][j] > *max1) {
52                *max1 = matriz1[i][j];
53            }
54        }
55    }
56 }
```

Ingrese el número de filas (n): 2
Ingrese el número de columnas (m): 3
Ingrese los elementos de la primera matriz:
Elemento [0][0]: 2
Elemento [0][1]: 2
Elemento [0][2]: 6
Elemento [1][0]: 5
Elemento [1][1]: 4
Elemento [1][2]: 8
Ingrese los elementos de la segunda matriz:
Elemento [0][0]: 9
Elemento [0][1]: 8
Elemento [0][2]: 4
Elemento [1][0]: 3
Elemento [1][1]: 4
Elemento [1][2]: 5
Máximo de la primera matriz: 6
Máximo de la segunda matriz: 9
Suma de los máximos: 14
Program finished with exit code 0
Press ENTER to exit console.

8. Que genere un solo arreglo a partir de dos arreglos iniciales, intercalando sus valores



```
1
2
3
4
5
6
7 int main() {
8     int arreglo1[MAX_SIZE], arreglo2[MAX_SIZE], arreglo3[MAX_SIZE * 2];
9     int n1, n2;
10
11    // Leer tamaño del primer arreglo
12    printf("Ingrese el tamaño del primer arreglo (n1): ");
13    scanf("%d", &n1);
14
15    // Leer elementos del primer arreglo
16    printf("Ingrese los elementos del primer arreglo:\n");
17    for (int i = 0; i < n1; i++) {
18        printf("Elemento %d: ", i + 1);
19        scanf("%d", &arreglo1[i]);
20    }
21
22    // Leer tamaño del segundo arreglo
23    printf("Ingrese el tamaño del segundo arreglo (n2): ");
24    scanf("%d", &n2);
25
26    // Leer elementos del segundo arreglo
27    printf("Ingrese los elementos del segundo arreglo:\n");
28    for (int i = 0; i < n2; i++) {
29        printf("Elemento %d: ", i + 1);
30        scanf("%d", &arreglo2[i]);
31    }
32
33    // Intercalar los arreglos
34    intercalarArreglos(arreglo1, arreglo2, arreglo3, n1, n2);
35
36    // Mostrar el arreglo resultante
37    printf("Arreglo intercalado:\n");
38    for (int i = 0; i < n1 + n2; i++) {
39        printf("%d ", arreglo3[i]);
40    }
41    printf("\n");
42
43    return 0;
44 }
45
46 void intercalarArreglos(int arreglo1[], int arreglo2[], int arreglo3[], int n1, int n2) {
47     int i, j, k;
48     i = j = k = 0;
49
50     while (i < n1 && j < n2) {
51         arreglo3[k++] = arreglo1[i++];
52         arreglo3[k++] = arreglo2[j++];
53     }
54
55     // Agregar los elementos restantes del primer arreglo, si hay
56     while (i < n1) {
57         arreglo3[k++] = arreglo1[i++];
58     }
59
60     // Agregar los elementos restantes del segundo arreglo, si hay
61     while (j < n2) {
62         arreglo3[k++] = arreglo2[j++];
63     }
64 }
```

Ingrese el tamaño del primer arreglo (n1): 4
Ingrese los elementos del primer arreglo:
Elemento 1: 4
Elemento 2: 3
Elemento 3: 3
Elemento 4: 3
Ingrese el tamaño del segundo arreglo (n2): 4
Ingrese los elementos del segundo arreglo:
Elemento 1: 2
Elemento 2: 3
Elemento 3: 4
Elemento 4: 2
Arreglo intercalado:
4 2 3 3 4 3 2
Program finished with exit code 0
Press ENTER to exit console.

OBSERVACIONES

¿QUÉ SE ME DIFICULTÓ?

Durante la práctica tuve dificultades al momento de desarrollar los problemas, puesto que en algunos ejercicios como el número 17, me costaba más el poder relacionar los bucles que necesitaba para poder llevar a cabo un correcto pseudocódigo. Por otra parte, también tuve dificultades al momento de desarrollar mis pseudocódigos puesto que no declaraba las variables suficientes para que se ejecutara mi programa.

¿CÓMO LO RESOLVÍ?

El ejercicio número 17 lo pude resolver con ayuda de algunos compañeros al analizar cómo ellos habían desarrollado su pseudocódigo, también me apoyé de ejercicios hechos en clase para poder ejecutar mi programa.

CONCLUSIÓN

Pude cumplir con los objetivos de la práctica y realizar los problemas planteados en clase. Se desarrollaron ejercicios de arreglos bidimensionales los cuales tienen una fila y una columna, siendo estos funcionales para poder hacer tablas. Por otra parte, pude resolver algunas dudas que tenía respecto a la ejecución de los mismos, puesto que aún no me quedaba tan claro cómo funcionaban este tipo de arreglos.