ejercicio 1:

```c
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

#define FILAS 4
#define COLUMNAS 4

// Función para generar valores aleatorios entre 0 y 9
int generarValorAleatorio() {
    return rand() % 10;
}

// Función para inicializar una matriz con valores aleatorios
void inicializarMatriz(int matriz[FILAS][COLUMNAS]) {
    for (int i = 0; i < FILAS; i++) {
        for (int j = 0; j < COLUMNAS; j++) {
            matriz[i][j] = generarValorAleatorio();
        }
    }
}

// Función para mostrar la matriz por pantalla
void mostrarMatriz(int matriz[FILAS][COLUMNAS]) {
    printf("Matriz generada:\n");
    for (int i = 0; i < FILAS; i++) {
        for (int j = 0; j < COLUMNAS; j++) {
            printf("%d\t", matriz[i][j]);
        }
        printf("\n");
    }
}

// Función para contar la cantidad de ceros en la matriz
int contarCeros(int matriz[FILAS][COLUMNAS]) {
    int contadorCeros = 0;
    for (int i = 0; i < FILAS; i++) {
        for (int j = 0; j < COLUMNAS; j++) {
            if (matriz[i][j] == 0) {
                contadorCeros++;
            }
        }
    }
    return contadorCeros;
}

int main() {
    // Inicializar la semilla para generar valores aleatorios
    srand(time(NULL));

    // Declaración de la matriz
    int matriz[FILAS][COLUMNAS];
```

```
    // Inicializar y mostrar la matriz
    inicializarMatriz(matriz);
    mostrarMatriz(matriz);

    // Contar la cantidad de ceros
    int cantidadCeros = contarCeros(matriz);

    // Mostrar la cantidad de ceros
    printf("\nCantidad de ceros en la matriz: %d\n", cantidadCeros);

    return 0;
}
```

ejercicio 2:

```
#include <stdio.h>

#define CHOFERES 20
#define DIAS_SEMANA 7

// Función para capturar los kilómetros recorridos por cada chofer
void capturarKilometros(int matriz[CHOFERES][DIAS_SEMANA]) {
    for (int i = 0; i < CHOFERES; i++) {
        printf("Ingrese los kilómetros recorridos por el chofer %d (legajo %d) durante la semana:\n", i + 1, i
+ 1);
        for (int j = 0; j < DIAS_SEMANA; j++) {
            printf("Día %d: ", j + 1);
            scanf("%d", &matriz[i][j]);
        }
    }
}

// Función para generar el informe con la cantidad de kilómetros por día y el total semanal
void generarInforme(int matriz[CHOFERES][DIAS_SEMANA]) {
    printf("\nInforme de kilómetros recorridos:\n");

    // Mostrar la cantidad de kilómetros por día y calcular el total semanal
    int totalSemanal = 0;
    for (int j = 0; j < DIAS_SEMANA; j++) {
        int totalDia = 0;
        printf("Día %d:\n", j + 1);

        for (int i = 0; i < CHOFERES; i++) {
            printf("  Chofer %d (legajo %d): %d km\n", i + 1, i + 1, matriz[i][j]);
            totalDia += matriz[i][j];
        }

        printf("  Total kilómetros en el día: %d km\n", totalDia);
        totalSemanal += totalDia;
    }

    // Mostrar el total semanal
```

```
    printf("\nTotal kilómetros durante la semana: %d km\n", totalSemanal);
}

int main() {
    // Declaración de la matriz para almacenar los kilómetros recorridos
    int matrizKilometros[CHOFERES][DIAS_SEMANA];

    // Capturar la información de los kilómetros recorridos
    capturarKilometros(matrizKilometros);

    // Generar el informe
    generarInforme(matrizKilometros);

    return 0;
}
```

ejercicio 3:

```
#include <stdio.h>

#define EMPLEADOS 5
#define DIAS_SEMANA 5

// Función para capturar las ventas de cada empleado
void capturarVentas(int matriz[EMPLEADOS][DIAS_SEMANA]) {
    for (int i = 0; i < EMPLEADOS; i++) {
        printf("Ingrese las ventas del empleado %d durante la semana:\n", i + 1);
        for (int j = 0; j < DIAS_SEMANA; j++) {
            printf("Día %d: $", j + 1);
            scanf("%d", &matriz[i][j]);
        }
    }
}

// Función para determinar la venta mayor por empleado
void determinarVentaMayor(int matriz[EMPLEADOS][DIAS_SEMANA]) {
    printf("\nVentas mayores por empleado:\n");

    for (int i = 0; i < EMPLEADOS; i++) {
        int ventaMayor = matriz[i][0];
        int diaVentaMayor = 1;

        for (int j = 1; j < DIAS_SEMANA; j++) {
            if (matriz[i][j] > ventaMayor) {
                ventaMayor = matriz[i][j];
                diaVentaMayor = j + 1;
            }
        }

        printf("Empleado %d: Venta mayor de $%d fue el día %d\n", i + 1, ventaMayor, diaVentaMayor);
    }
```

```
}

int main() {
    // Declaración de la matriz para almacenar las ventas
    int matrizVentas[EMPLEADOS][DIAS_SEMANA];

    // Capturar la información de las ventas
    capturarVentas(matrizVentas);

    // Determinar la venta mayor por empleado
    determinarVentaMayor(matrizVentas);

    return 0;
}
```

ejercicio 4:

```
#include <stdio.h>

#define MAX_ELEMENTOS 10

// Función para leer un vector
void leerVector(int vector[], int longitud) {
    printf("Ingrese los elementos del vector:\n");
    for (int i = 0; i < longitud; i++) {
        printf("Elemento %d: ", i + 1);
        scanf("%d", &vector[i]);
    }
}

// Función para rotar el vector
void rotarVector(int vector[], int longitud) {
    int temp = vector[longitud - 1]; // Almacena el último elemento

    // Desplazar los elementos hacia la derecha
    for (int i = longitud - 1; i > 0; i--) {
        vector[i] = vector[i - 1];
    }

    // Asignar el último elemento al primer lugar
    vector[0] = temp;
}

// Función para mostrar el vector
void mostrarVector(int vector[], int longitud) {
    printf("Vector resultante:\n");
    for (int i = 0; i < longitud; i++) {
        printf("%d ", vector[i]);
    }
    printf("\n");
}
```

```
int main() {
    // Declaración de variables
    int vector[MAX_ELEMENTOS];
    int longitud;

    // Ingreso de la longitud del vector (asegurarse de que sea menor o igual a MAX_ELEMENTOS)
    printf("Ingrese la longitud del vector (menor o igual a %d): ", MAX_ELEMENTOS);
    scanf("%d", &longitud);

    // Verificar que la longitud sea válida
    if (longitud <= 0 || longitud > MAX_ELEMENTOS) {
        printf("Longitud no válida. El programa terminará.\n");
        return 1;
    }

    // Leer el vector
    leerVector(vector, longitud);

    // Rotar el vector
    rotarVector(vector, longitud);

    // Mostrar el vector resultante
    mostrarVector(vector, longitud);

    return 0;
}
```

ejercicio 5:

```
#include <stdio.h>

#define FILAS 15
#define COLUMNAS 12

// Función para leer el arreglo
void leerArreglo(int arreglo[FILAS][COLUMNAS]) {
    printf("Ingrese los elementos del arreglo:\n");
    for (int i = 0; i < FILAS; i++) {
        for (int j = 0; j < COLUMNAS; j++) {
            printf("Elemento [%d][%d]: ", i + 1, j + 1);
            scanf("%d", &arreglo[i][j]);
        }
    }
}

// Función para encontrar el menor elemento del arreglo
int encontrarMenorElemento(int arreglo[FILAS][COLUMNAS]) {
    int menor = arreglo[0][0];

    for (int i = 0; i < FILAS; i++) {
```

```
        for (int j = 0; j < COLUMNAS; j++) {
            if (arreglo[i][j] < menor) {
                menor = arreglo[i][j];
            }
        }
    }

    return menor;
}

// Función para calcular la suma de los elementos de las cinco primeras filas
int calcularSumaFilas(int arreglo[FILAS][COLUMNAS]) {
    int suma = 0;

    for (int i = 0; i < 5; i++) {
        for (int j = 0; j < COLUMNAS; j++) {
            suma += arreglo[i][j];
        }
    }

    return suma;
}

// Función para contar los elementos negativos en las columnas de la quinta a la novena
int contarNegativosColumnas(int arreglo[FILAS][COLUMNAS]) {
    int contadorNegativos = 0;

    for (int i = 0; i < FILAS; i++) {
        for (int j = 4; j < 9; j++) {
            if (arreglo[i][j] < 0) {
                contadorNegativos++;
            }
        }
    }

    return contadorNegativos;
}

int main() {
    // Declaración del arreglo
    int arreglo[FILAS][COLUMNAS];

    // Leer el arreglo
    leerArreglo(arreglo);

    // Encontrar el menor elemento del arreglo
    int menorElemento = encontrarMenorElemento(arreglo);
    printf("\nEl menor elemento del arreglo es: %d\n", menorElemento);

    // Calcular la suma de los elementos de las cinco primeras filas
    int sumaFilas = calcularSumaFilas(arreglo);
    printf("La suma de los elementos de las cinco primeras filas es: %d\n", sumaFilas);
```

```
    // Contar los elementos negativos en las columnas de la quinta a la novena
    int negativosColumnas = contarNegativosColumnas(arreglo);
    printf("El total de elementos negativos en las columnas de la quinta a la novena es: %d\n",
negativosColumnas);

    return 0;
}
```

ejercicio 6:

```
#include <stdio.h>

#define ORDEN_MATRIZ 12

// Función para leer una matriz cuadrada
void leerMatrizCuadrada(int matriz[ORDEN_MATRIZ][ORDEN_MATRIZ]) {
    printf("Ingrese los elementos de la matriz cuadrada:\n");
    for (int i = 0; i < ORDEN_MATRIZ; i++) {
        for (int j = 0; j < ORDEN_MATRIZ; j++) {
            printf("Elemento [%d][%d]: ", i + 1, j + 1);
            scanf("%d", &matriz[i][j]);
        }
    }
}

// Función para determinar si la diagonal principal de dos matrices es igual
int diagonalPrincipalIgual(int matriz1[ORDEN_MATRIZ][ORDEN_MATRIZ], int
matriz2[ORDEN_MATRIZ][ORDEN_MATRIZ]) {
    for (int i = 0; i < ORDEN_MATRIZ; i++) {
        if (matriz1[i][i] != matriz2[i][i]) {
            return 0; // La diagonal principal no es igual
        }
    }
    return 1; // La diagonal principal es igual
}

int main() {
    // Declaración de las matrices cuadradas
    int matriz1[ORDEN_MATRIZ][ORDEN_MATRIZ];
    int matriz2[ORDEN_MATRIZ][ORDEN_MATRIZ];

    // Leer las matrices cuadradas
    leerMatrizCuadrada(matriz1);
    leerMatrizCuadrada(matriz2);

    // Determinar si la diagonal principal de las matrices es igual
    if (diagonalPrincipalIgual(matriz1, matriz2)) {
        printf("\nLa diagonal principal de las matrices es igual.\n");
    } else {
        printf("\nLa diagonal principal de las matrices no es igual.\n");
```

```
    }

    return 0;
}
```

ejercicio 7:

```
#include <stdio.h>

#define FILAS 12
#define COLUMNAS 19

// Función para leer una matriz
void leerMatriz(int matriz[FILAS][COLUMNAS]) {
    printf("Ingrese los elementos de la matriz:\n");
    for (int i = 0; i < FILAS; i++) {
        for (int j = 0; j < COLUMNAS; j++) {
            printf("Elemento [%d][%d]: ", i + 1, j + 1);
            scanf("%d", &matriz[i][j]);
        }
    }
}

// Función para cambiar los elementos negativos por ceros en una matriz
void cambiarNegativosACeros(int matriz[FILAS][COLUMNAS]) {
    for (int i = 0; i < FILAS; i++) {
        for (int j = 0; j < COLUMNAS; j++) {
            if (matriz[i][j] < 0) {
                matriz[i][j] = 0;
            }
        }
    }
}

// Función para mostrar una matriz
void mostrarMatriz(int matriz[FILAS][COLUMNAS]) {
    printf("Matriz resultante:\n");
    for (int i = 0; i < FILAS; i++) {
        for (int j = 0; j < COLUMNAS; j++) {
            printf("%d\t", matriz[i][j]);
        }
        printf("\n");
    }
}

int main() {
    // Declaración de la matriz
    int matriz[FILAS][COLUMNAS];

    // Leer la matriz
    leerMatriz(matriz);
```

```
    // Cambiar los elementos negativos por ceros
    cambiarNegativosACeros(matriz);

    // Mostrar la matriz resultante
    mostrarMatriz(matriz);

    return 0;
}
```

ejercicio 8:

```
#include <stdio.h>

#define FILAS 5
#define COLUMNAS 6

// Función para leer una matriz
void leerMatriz(int matriz[FILAS][COLUMNAS]) {
    printf("Ingrese los elementos de la matriz:\n");
    for (int i = 0; i < FILAS; i++) {
        for (int j = 0; j < COLUMNAS; j++) {
            printf("Elemento [%d][%d]: ", i + 1, j + 1);
            scanf("%d", &matriz[i][j]);
        }
    }
}

// Función para contar elementos negativos en una matriz
int contarNegativos(int matriz[FILAS][COLUMNAS]) {
    int contadorNegativos = 0;
    for (int i = 0; i < FILAS; i++) {
        for (int j = 0; j < COLUMNAS; j++) {
            if (matriz[i][j] < 0) {
                contadorNegativos++;
            }
        }
    }
    return contadorNegativos;
}

// Función para contar elementos de la diagonal principal que son igual a cero
int contarCerosDiagonalPrincipal(int matriz[FILAS][COLUMNAS]) {
    int contadorCeros = 0;
    for (int i = 0; i < FILAS; i++) {
        if (matriz[i][i] == 0) {
            contadorCeros++;
        }
    }
    return contadorCeros;
}
```

```
int main() {
    // Declaración de la matriz
    int matriz[FILAS][COLUMNAS];

    // Leer la matriz
    leerMatriz(matriz);

    // Contar elementos negativos en la matriz
    int negativos = contarNegativos(matriz);
    printf("\nEl total de elementos negativos en la matriz es: %d\n", negativos);

    // Contar elementos de la diagonal principal que son igual a cero
    int cerosDiagonalPrincipal = contarCerosDiagonalPrincipal(matriz);
    printf("El total de elementos de la diagonal principal que son igual a cero es: %d\n",
cerosDiagonalPrincipal);

    return 0;
}
```

ejercicio 9:

```
#include <stdio.h>
#include <string.h>

#define CHOEFERES 5
#define DIAS_SEMANA 6

// Estructura para almacenar la información de cada chofer
struct Chofer {
    int legajo;
    char nombre[50];
    float horasTrabajadas[DIAS_SEMANA];
    float sueldoPorHora;
};

// Función para calcular el total de horas trabajadas a la semana para cada chofer
void calcularTotalHoras(struct Chofer choferes[CHOEFERES]) {
    for (int i = 0; i < CHOEFERES; i++) {
        float totalHoras = 0;
        for (int j = 0; j < DIAS_SEMANA; j++) {
            totalHoras += choferes[i].horasTrabajadas[j];
        }
        printf("Total de horas trabajadas para %s: %.2f horas\n", choferes[i].nombre, totalHoras);
    }
}

// Función para calcular el sueldo semanal para cada chofer
void calcularSueldoSemanal(struct Chofer choferes[CHOEFERES]) {
    for (int i = 0; i < CHOEFERES; i++) {
        float totalHoras = 0;
```

```
        for (int j = 0; j < DIAS_SEMANA; j++) {
            totalHoras += choferes[i].horasTrabajadas[j];
        }
        float sueldoSemanal = totalHoras * choferes[i].sueldoPorHora;
        printf("Sueldo semanal para %s: $%.2f\n", choferes[i].nombre, sueldoSemanal);
    }
}

// Función para calcular el total que pagará la empresa
void calcularTotalEmpresa(struct Chofer choferes[CHOEFERES]) {
    float totalEmpresa = 0;
    for (int i = 0; i < CHOEFERES; i++) {
        float totalHoras = 0;
        for (int j = 0; j < DIAS_SEMANA; j++) {
            totalHoras += choferes[i].horasTrabajadas[j];
        }
        totalEmpresa += totalHoras * choferes[i].sueldoPorHora;
    }
    printf("Total que pagará la empresa: $%.2f\n", totalEmpresa);
}

// Función para indicar el nombre del trabajador que labora más horas el día lunes
void trabajadorMasHorasLunes(struct Chofer choferes[CHOEFERES]) {
    float maxHoras = 0;
    char nombre[50];
    for (int i = 0; i < CHOEFERES; i++) {
        if (choferes[i].horasTrabajadas[0] > maxHoras) {
            maxHoras = choferes[i].horasTrabajadas[0];
            strcpy(nombre, choferes[i].nombre);
        }
    }
    printf("El trabajador que labora más horas el día lunes es: %s\n", nombre);
}

// Función para imprimir el reporte con todos los datos
void imprimirReporte(struct Chofer choferes[CHOEFERES]) {
    printf("\n--- Reporte ---\n");
    calcularTotalHoras(choferes);
    calcularSueldoSemanal(choferes);
    calcularTotalEmpresa(choferes);
    trabajadorMasHorasLunes(choferes);
}

int main() {
    // Declaración de los choferes
    struct Chofer choferes[CHOEFERES] = {
        {1, "Juan", {8, 7, 6, 8, 7, 6}, 10.0},
        {2, "Ana", {9, 8, 7, 6, 5, 8}, 12.0},
        {3, "Pedro", {7, 7, 8, 6, 7, 8}, 11.0},
        {4, "María", {8, 7, 8, 7, 6, 5}, 10.5},
        {5, "Luis", {7, 6, 5, 6, 7, 8}, 11.5},
    };
```

```
    // Imprimir el reporte con todos los datos
    imprimirReporte(choferes);

    return 0;
}
```

ejercicio 10:

```
#include <stdio.h>

#define FILAS 10
#define COLUMNAS 10

// Función para leer una matriz
void leerMatriz(int matriz[FILAS][COLUMNAS]) {
    printf("Ingrese los elementos de la matriz:\n");
    for (int i = 0; i < FILAS; i++) {
        for (int j = 0; j < COLUMNAS; j++) {
            printf("Elemento [%d][%d]: ", i + 1, j + 1);
            scanf("%d", &matriz[i][j]);
        }
    }
}

// Función para calcular la suma de filas y columnas de una matriz
void calcularSumaFilasColumnas(int matriz[FILAS][COLUMNAS], int sumaFilas[FILAS], int
sumaColumnas[COLUMNAS]) {
    // Inicializar los vectores de suma en 0
    for (int i = 0; i < FILAS; i++) {
        sumaFilas[i] = 0;
    }
    for (int j = 0; j < COLUMNAS; j++) {
        sumaColumnas[j] = 0;
    }

    // Calcular la suma de filas y columnas
    for (int i = 0; i < FILAS; i++) {
        for (int j = 0; j < COLUMNAS; j++) {
            sumaFilas[i] += matriz[i][j];
            sumaColumnas[j] += matriz[i][j];
        }
    }
}

// Función para mostrar un vector
void mostrarVector(int vector[], int longitud, char nombre[]) {
    printf("%s: [", nombre);
    for (int i = 0; i < longitud; i++) {
        printf("%d", vector[i]);
        if (i < longitud - 1) {
```

```
            printf(", ");
        }
    }
    printf("]\n");
}

int main() {
    // Declaración de la matriz y los vectores de suma
    int matriz[FILAS][COLUMNAS];
    int sumaFilas[FILAS];
    int sumaColumnas[COLUMNAS];

    // Leer la matriz
    leerMatriz(matriz);

    // Calcular la suma de filas y columnas
    calcularSumaFilasColumnas(matriz, sumaFilas, sumaColumnas);

    // Mostrar los vectores de suma
    mostrarVector(sumaFilas, FILAS, "Suma de Filas");
    mostrarVector(sumaColumnas, COLUMNAS, "Suma de Columnas");

    return 0;
}
```

ejercicio 11:

```
#include <stdio.h>

#define TORRES 7
#define PISOS_POR_TORRE 20
#define DEPARTAMENTOS_POR_PISO 6

// Función para calcular la cantidad total de habitantes del complejo
int calcularCantidadTotalHabitantes(int
complejo[TORRES][PISOS_POR_TORRE][DEPARTAMENTOS_POR_PISO]) {
    int totalHabitantes = 0;
    for (int i = 0; i < TORRES; i++) {
        for (int j = 0; j < PISOS_POR_TORRE; j++) {
            for (int k = 0; k < DEPARTAMENTOS_POR_PISO; k++) {
                totalHabitantes++;
            }
        }
    }
    return totalHabitantes;
}

// Función para calcular la cantidad total de habitantes por torre
int calcularCantidadHabitantesPorTorre(int
complejo[TORRES][PISOS_POR_TORRE][DEPARTAMENTOS_POR_PISO]) {
    int totalHabitantesPorTorre[TORRES] = {0};
```

```
    for (int i = 0; i < TORRES; i++) {
        for (int j = 0; j < PISOS_POR_TORRE; j++) {
            for (int k = 0; k < DEPARTAMENTOS_POR_PISO; k++) {
                totalHabitantesPorTorre[i]++;
            }
        }
    }
    int totalHabitantes = 0;
    for (int i = 0; i < TORRES; i++) {
        totalHabitantes += totalHabitantesPorTorre[i];
        printf("Total de habitantes en la torre %d: %d\n", i + 1, totalHabitantesPorTorre[i]);
    }
    return totalHabitantes;
}

// Función para calcular la cantidad promedio de habitantes por torre
float calcularPromedioHabitantesPorTorre(int
complejo[TORRES][PISOS_POR_TORRE][DEPARTAMENTOS_POR_PISO]) {
    int totalHabitantes = calcularCantidadTotalHabitantes(complejo);
    int totalHabitantesPorTorre = calcularCantidadHabitantesPorTorre(complejo);
    return (float)totalHabitantesPorTorre / TORRES;
}

// Función para calcular la cantidad promedio de habitantes por piso
float calcularPromedioHabitantesPorPiso(int
complejo[TORRES][PISOS_POR_TORRE][DEPARTAMENTOS_POR_PISO]) {
    int totalHabitantes = calcularCantidadTotalHabitantes(complejo);
    return (float)totalHabitantes / (TORRES * PISOS_POR_TORRE);
}

int main() {
    // Declaración del complejo de torres como una matriz
    int complejo[TORRES][PISOS_POR_TORRE][DEPARTAMENTOS_POR_PISO];

    // Inicialización del complejo de torres (puedes cargar datos de habitantes si lo deseas)

    // Calcular e imprimir los resultados
    int totalHabitantes = calcularCantidadTotalHabitantes(complejo);
    printf("\nCantidad total de habitantes del complejo: %d\n", totalHabitantes);

    int totalHabitantesPorTorre = calcularCantidadHabitantesPorTorre(complejo);
    printf("\nCantidad promedio de habitantes por torre: %.2f\n",
calcularPromedioHabitantesPorTorre(complejo));

    printf("\nCantidad promedio de habitantes por piso: %.2f\n",
calcularPromedioHabitantesPorPiso(complejo));

    return 0;
}
```

ejercicio 12:

```
#include <stdio.h>

#define SEMANAS 4
#define DIAS_POR_SEMANA 7

// Función para leer las ventas diarias de cada semana
void leerVentas(int ventas[SEMANAS][DIAS_POR_SEMANA]) {
    printf("Ingrese las ventas diarias de cada semana:\n");
    for (int i = 0; i < SEMANAS; i++) {
        printf("Semana %d:\n", i + 1);
        for (int j = 0; j < DIAS_POR_SEMANA; j++) {
            printf("Día %d: ", j + 1);
            scanf("%d", &ventas[i][j]);
        }
    }
}

// Función para calcular el total de ventas de cada semana
void calcularTotalVentas(int ventas[SEMANAS][DIAS_POR_SEMANA], int totalVentas[SEMANAS]) {
    for (int i = 0; i < SEMANAS; i++) {
        totalVentas[i] = 0;
        for (int j = 0; j < DIAS_POR_SEMANA; j++) {
            totalVentas[i] += ventas[i][j];
        }
    }
}

// Función para calcular el promedio de ventas de cada semana
void calcularPromedioVentas(int totalVentas[SEMANAS], float promedioVentas[SEMANAS]) {
    for (int i = 0; i < SEMANAS; i++) {
        promedioVentas[i] = (float)totalVentas[i] / DIAS_POR_SEMANA;
    }
}

// Función para determinar la semana con la mayor venta
int determinarSemanaMayorVenta(int totalVentas[SEMANAS]) {
    int semanaMayorVenta = 0;
    for (int i = 1; i < SEMANAS; i++) {
        if (totalVentas[i] > totalVentas[semanaMayorVenta]) {
            semanaMayorVenta = i;
        }
    }
    return semanaMayorVenta;
}

// Función para mostrar un vector
void mostrarVector(int vector[], int longitud, char nombre[]) {
    printf("%s: [", nombre);
    for (int i = 0; i < longitud; i++) {
        printf("%d", vector[i]);
        if (i < longitud - 1) {
```

```
            printf(", ");
        }
    }
    printf("]\n");
}

int main() {
    // Declaración de la matriz de ventas y los vectores de resultados
    int ventas[SEMANAS][DIAS_POR_SEMANA];
    int totalVentas[SEMANAS];
    float promedioVentas[SEMANAS];

    // Leer las ventas diarias
    leerVentas(ventas);

    // Calcular el total de ventas de cada semana
    calcularTotalVentas(ventas, totalVentas);

    // Calcular el promedio de ventas de cada semana
    calcularPromedioVentas(totalVentas, promedioVentas);

    // Determinar la semana con la mayor venta
    int semanaMayorVenta = determinarSemanaMayorVenta(totalVentas);

    // Mostrar los resultados
    mostrarVector(totalVentas, SEMANAS, "Total de Ventas");
    mostrarVector(promedioVentas, SEMANAS, "Promedio de Ventas");
    printf("La semana con la mayor venta fue la semana %d\n", semanaMayorVenta + 1);

    return 0;
}
```