DOCUMENT_HEADING

3 files

excension de responsabilidad.c
funcion.c
pivotajeparcialescalonado (1).c

excension de responsabilidad.c

```
/*

Exención de responsabilidad:

Al descargar y/o utilizar los materiales proporcionados en este archivo (en adelante, "los Materiales"), el usuario (en adelante, "el Usuario") acepta los términos y condiciones que se describen a continuación:

Los Materiales son proporcionados "tal cual" y sin garantías de ningún tipo, ya sean expresas o implícitas. El autor de los Materiales no garantiza la precisión, exhaustividad, actualidad o idoneidad de los mismos para un propósito particular.

Los Materiales se ofrecen únicamente como refuerzo o ayuda para realizar más ejercicios, y no están destinados a ser copiados directamente. El Usuario debe utilizarlos como una guía o recurso de apoyo en su aprendizaje y no como una solución completa para sus tareas o trabajos académicos.

El autor de los Materiales no se hace responsable de ningún error, omisión, inexactitud o malentendido en la información proporcionada.

El Usuario acepta asumir todos los riesgos asociados con la utilización de los Materiales y será el único responsable de cualquier daño, pérdida, perjuicio o inconveniente que pueda surgir como resultado del uso o la incapacidad de usar los Materiales.

El Usuario se compromete a no responsabilizar al autor de los Materiales por cualquier reclamo, demanda, acción, responsabilidad, costo o gasto, incluidos los honorarios legales, que surjan de o estén relacionados con el uso o la dependencia de los Materiales.

Los Materiales no deben ser utilizados como sustituto del asesoramiento, la supervisión o la instrucción de un profesor, tutor u otro profesional cualificado en la materia.

El Usuario no debe compartir, distribuir, modificar, vender, transmitir, copiar o reproducir en cualquier forma, total o parcialmente, los Materiales sin la previa autorización por escrito del autor.

Al descargar y/o utilizar los Materiales, el Usuario confirma que ha leído, comprendido y aceptado los términos y condiciones aquí establecidos.


*/
```

funcion.c

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
```

```
// Función que hace la salida de la matriz final
void imprimirMatriz(double **A, int n)
{
    int i, j;
    for (i = 0; i < n; i++)
    {
        for (j = 0; j < n; j++)
        {
            printf("%.2lf ", A[i][j]);
        }
        printf("\n");
    }
}
/*
La función main pide al usuario el tamaño
de la matriz cuadrada (este paso se puede omitir por una variable global fija)
y asignando memoria dinámica para esta matriz y un arreglo llamado "scale".
Luego se pide al usuario que ingrese los valores de la matriz.

El siguiente bucle for calcula el valor máximo de cada fila de la
matriz A y lo almacena.

Hay un bucle for anidado que se encarga de encontrar el elemento
mayor de la columna y cambia su correspondiente fila con k ( la fila k es solo si no está previamente ),

El siguiente for es para convertir la matriz A en la matriz.

Usamos el "free(scale);" para liberar la memoria de la variable
y despúes hacemos la llamada a la funcion que imprime la matriz A.
*/
int main()
{
    int n, i, j, k, maxI, temp;
    double maxVal, tempf, **A;
    double *scale;

    printf("Ingrese el tamaño de la matriz cuadrada: ");
    scanf("%d", &n);
    // memoria dinámica matriz A
    A = (double **)malloc(n * sizeof(double *));
    for (i = 0; i < n; i++)
    {
        A[i] = (double *)malloc(n * sizeof(double));
    }
    scale = (double *)malloc(n * sizeof(double));
    // valores matriz A
    printf("Ingrese los valores de la matriz:\n");
    for (i = 0; i < n; i++)
    {
        for (j = 0; j < n; j++)
        {
            scanf("%lf", &A[i][j]);
        }
    }

    for (i = 0; i < n; i++)
    {
        scale[i] = 0;
```

```
            for (j = 0; j < n; j++)
            {
                if (scale[i] < fabs(A[i][j]))
                    scale[i] = fabs(A[i][j]);
            }
        }

        for (k = 0; k < n - 1; k++)
        {
            maxI = k;
            maxVal = fabs(A[k][k] / scale[k]);
            for (i = k + 1; i < n; i++)
            {
                tempf = fabs(A[i][k] / scale[i]);
                if (tempf > maxVal)
                {
                    maxVal = tempf;
                    maxI = i;
                }
            }

            if (maxI != k)
            {
                for (j = 0; j < n; j++)
                {
                    tempf = A[k][j];
                    A[k][j] = A[maxI][j];
                    A[maxI][j] = tempf;
                }
                temp = scale[k];
                scale[k] = scale[maxI];
                scale[maxI] = temp;
            }
            for (i = k + 1; i < n; i++)
            {
                A[i][k] = A[i][k] / A[k][k];
                for (j = k + 1; j < n; j++)
                {
                    A[i][j] = A[i][j] - A[i][k] * A[k][j];
                }
            }
        }

        free(scale);
        imprimirMatriz(A, n);

        return 0;
    }
```

pivotajeparcialescalonado (1).c

```
#include <stdio.h>

#define N 4 // se define una constante N con un valor de 4

void imprimir_matriz(double matriz[N][N])
{ // se define una función para imprimir la matriz con un formato específico
```

```
    for (int i = 0; i < N; i++)
    {
        for (int j = 0; j < N; j++)
        {
            printf("%.2f\t", matriz[i][j]); // se imprime cada elemento de la
matriz con dos decimales
        }
        printf("\n"); // salto de línea después de cada fila
    }
}

int main()
{
    double matriz[N][N]; // se declara la matriz 4x4
    double escalar[N];   // se declara un arreglo escalar de tamaño N
    int pivote[N];       // se declara un arreglo pivote de tamaño N

    printf("Introduce los elementos de la matriz 4x4:\n");
    for (int i = 0; i < N; i++)
    { // ciclo para recibir los elementos de la matriz
        for (int j = 0; j < N; j++)
        {
            if (scanf("%lf", &matriz[i][j]) != 1)
            { // se verifica si la matriz es 4x4
                printf("La matriz introducida no es 4x4, por favor vuelve a
intentarlo.\n");
                return 0;
            }
        }
    }

    for (int i = 0; i < N; i++)
    {                               // ciclo para asignar el escalar y el
pivote a cada fila de la matriz
        escalar[i] = matriz[i][0]; // se asigna el primer elemento de cada
fila como escalar
        pivote[i] = i;             // se asigna el índice de cada fila como
pivote
        for (int j = 1; j < N; j++)
        {
            if (escalar[i] < matriz[i][j])
            { // se busca el valor máximo de cada fila
                escalar[i] = matriz[i][j];
            }
        }
    }

    for (int i = 0; i < N - 1; i++)
    { // ciclo para escalar la matriz utilizando el método de Gauss-Jordan
        double max_ratio = 0;
        int max_ratio_row = i;
        for (int j = i; j < N; j++)
        {
            double ratio = matriz[pivote[j]][i] / escalar[pivote[j]];
            if (ratio > max_ratio)
            { // se busca el máximo ratio entre el elemento y el escalar
                max_ratio = ratio;
                max_ratio_row = j;
            }
        }
```

```
            int temp = pivote[i]; // se intercambian los valores de los índices
            pivote[i] = pivote[max_ratio_row];
            pivote[max_ratio_row] = temp;

            for (int j = i + 1; j < N; j++) // se realiza un ciclo para
actualizar los valores de la matriz utilizando Gauss-Jordan.
            {
                double factor = matriz[pivote[j]][i] / matriz[pivote[i]][i];
                matriz[pivote[j]][i] = factor;
                for (int k = i + 1; k < N; k++)
                {
                    matriz[pivote[j]][k] = matriz[pivote[j]][k] - factor *
matriz[pivote[i]][k];
                }
            }
        }

        printf("Matriz escalada:\n"); // se imprime la matriz escalada y se
finaliza la ejecucion del programa
        imprimir_matriz(matriz);
        return 0;
    }
```