	Carátula para entrega de prácticas
Facultad de Ingeniería	Laboratorio de docencia

Laboratorio de computación Salas A y B

Profesor: Dr. Ismael Everardo Bárcenas Patiño

Asignatura: Estructuras de datos y algoritmos I

Grupo: 3

No. de práctica: 4

Integrantes:
10 Gurrión Aquino Carlos Ángel
14 León Ruiz Eduardo
16 Macías Niño Carmen Violeta
17 Marroquín García Ricardo
21 Montaño Torres Rodolfo Santiago

No. de equipo de cómputo empleado:

No. de brigada: 7

Semestre: 2022-1

Fecha de entrega: 28 de septiembre de 2021

Observaciones:

Calificación:

Rúbrica de evaluación:

100	El programa cumple con todos los requerimientos
80-99	El programa cumple con la mayoría de los requerimientos
60-79	El programa cumple con algunos de los requerimientos
50-59	La lógica del programa es correcta, pero no corre o se cuelga

Práctica 4 | Almacenamiento en tiempo de ejecución

Problema a resolver:

I. Escribir un programa en lenguaje C que cumpla con las siguientes características:

- calcule el producto de dos matrices cuyos elementos sean números enteros;
- las matrices estén implementadas en estructuras de datos dinámicas;
- se pueda repetir la función de multiplicación de matrices: modificar la memoria reservada para las estructuras de datos correspondientes a las matrices.

Código solución:

Enlace al archivo C guardado en nuestro repositorio de GitHub:
https://github.com/Rodolfo-Santiago/EDA1-2022-1-Practicas/blob/main/codes/eda1_p4_code.c

Enlace al código en un compilador en línea:
https://onlinegdb.com/taGID4XTy

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


int** matriz_A;
int** matriz_B;

int filas_A;
int columnas_A;
int filas_B;
int columnas_B;

int cont_x;
int cont_y;

char letra_A = 'A';
char letra_B = 'B';


void descripcion_programa( );

int** creacion_de_una_matriz(int**, int*, int*, char*);
void lectura_de_filas_columnas(int*);
int** asignacion_de_espacio_en_memoria(int**, int*, int*);
void insercion_de_numeros_dentro_de_matriz(int**, int*, int*);

void impresion_de_matriz(int**, int*, int*, char*);

void multiplicacion_de_matrices( );
void liberar_filas_de_matriz(int**, int*);



int main()
{
      int opcion;

      descripcion_programa( );

      while (1)
      {
            filas_B = 0;


            matriz_A = creacion_de_una_matriz( matriz_A, &filas_A, &columnas_A, &letra_A );
            matriz_B = creacion_de_una_matriz( matriz_B, &filas_B, &columnas_B, &letra_B );


            impresion_de_matriz( matriz_A, &filas_A, &columnas_A, &letra_A );
            impresion_de_matriz( matriz_B, &filas_B, &columnas_B, &letra_B );
```

```
		multiplicacion_de_matrices( );



		printf("\n\nSi desea salir ingrese 0. Si no, cualquier otro entero.\n");
		scanf("%d", &opcion);

		if (opcion == 0)
		{
			free(matriz_A);
			free(matriz_B);

			return 0;

		}
	}

	return 0;
}



void descripcion_programa( )
{
	printf("PROGRAMA PARA CALCULAR EL PRODUCTO DE DOS MATRICES\n\n");
	printf("Este programa recibe como entrada dos matrices A y B de tamaño m x n y n x p");
	printf(" respectivamente (donde m, n, p son enteros positivos) cuyos elementos son ");
	printf("numeros enteros. El programa multiplica ambas matrices A y B.  Y como datos de");
	printf(" salida se imprimen las matrices factores y la matriz producto.\n");
}



int** creacion_de_una_matriz( int** matriz_previa, int* filas, int* columnas, char* caracter )
{
	int** matriz;

	printf("\n\nIngrese el tamaño de la matriz %c:\n", *caracter);

	printf("\tNumero de filas: ");
	lectura_de_filas_columnas( filas );
	printf ("\tNumero de columnas: ");
	lectura_de_filas_columnas( columnas );


	matriz = asignacion_de_espacio_en_memoria( matriz_previa, filas, columnas );


	insercion_de_numeros_dentro_de_matriz( matriz, filas, columnas );

	return matriz;
}



void lectura_de_filas_columnas( int* dimension )
{
	scanf("%d", dimension);


	while (filas_B != 0 && filas_B != columnas_A)
	{
		printf("\tAdvertencia: las filas de B deben ser iguales que las columnas de A.\n");
```

```
		scanf("%d", &filas_B);
	}

	while (*dimension < 1)
	{
		printf("\tAdvertencia: el valor debe ser un entero positivo.\n");
		scanf("%d", dimension);
	}
}



int** asignacion_de_espacio_en_memoria( int** matriz_previa, int* filas, int* columnas )
{
	int** matriz_nueva;


	matriz_nueva = (int**) realloc(matriz_previa, (*filas)*sizeof(int*) );

	for (cont_x = 0; cont_x < *filas; cont_x++)
	{

		*(matriz_nueva + cont_x) = (int*) calloc(*columnas, sizeof(int) );
	}

	return matriz_nueva;
}



void insercion_de_numeros_dentro_de_matriz( int** matriz, int* filas, int* columnas )
{
	printf("\n\tIngrese los %d elementos de la matriz fila por fila.\n", (*filas) * (*columnas) );

	for (cont_x = 0; cont_x < *filas; cont_x++)
	{
		printf("\tIngrese los %d enteros de la fila %d:\n", *columnas, cont_x + 1);
		for (cont_y = 0; cont_y < *columnas; cont_y++)
		{
			printf("\t%d,%d: ", cont_x + 1, cont_y + 1);
			scanf("%d", *(matriz + cont_x) + cont_y );
		}
	}
}



void impresion_de_matriz( int** matriz, int* filas, int* columnas, char* caracter )
{
	printf("\n\tMatriz %c:\n", *caracter);

	for (cont_x = 0; cont_x < *filas; cont_x++)
	{
		printf("\t\t\t");
		for (cont_y = 0; cont_y < *columnas; cont_y++)
		{
			printf("%5d  ", *(*(matriz + cont_x) + cont_y) );
		}
		printf("\n");
	}
	printf("\n");
}
```

```
void multiplicacion_de_matrices( )
{
      int cont_z;
      int resultado;

      printf("\n\tMatriz A x B:\n");

      for (cont_x = 0; cont_x < filas_A; cont_x++)
      {
             printf("\t\t\t");

             for (cont_y = 0; cont_y < columnas_B; cont_y++)
             {
                    for (resultado = 0, cont_z = 0; cont_z < columnas_A; cont_z++)
                    {

                    resultado += ( *(*(matriz_A + cont_x) + cont_z) ) * ( *(*(matriz_B + cont_z) + cont_y) );
                    }

                    printf("%5d  ", resultado);
             }

             printf("\n");
      }


      liberar_filas_de_matriz( matriz_A, &filas_A );
      liberar_filas_de_matriz( matriz_B, &filas_B );
}



void liberar_filas_de_matriz( int** matriz, int* filas )
{
      for (cont_x = 0; cont_x < *filas; cont_x++)
      {
             free( *( matriz + cont_x ) );
      }
}
```