	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: 3

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: 21 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 3 | Tipo de dato abstracto

Problema a resolver:

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

- Una base de datos de películas implementada en un arreglo.
- Los registros de la base de datos implementada en una estructura de datos abtracta, con los siguientes campos:
  - Identificador único.
  - Nombre de la película.
  - Año de realización.
  - Género.
  - Calificación: 0-10.
- Una función para agregar películas a la base de datos.
- Una función para imprimir todas las películas de la base de datos.
- Una función para buscar, a partir del identificador, una película en la base de datos.

Código solución:

Enlace al código en compilador en línea: https://www.onlinegdb.com/edit/kdRtBvhTI

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

// Declaración de variables globales
int i;

// Declaración de estructuras de datos abstractas
struct MovieProperties  // Estructura para guardar las propiedades de las películas
{
      int id;
      char name[50];
      int date;
      char genre[20];
      float score;
};

struct ListOfMovies  // Estructura en forma de lista ligada, para guardar las películas
{
      struct MovieProperties movieX;  // Estructura MovieProperties dentro de la estructura ListOfMovies
      struct ListOfMovies* next;  // Apuntador a otra estructura
};

// Declaración de funciones
void insertOneMovie(struct ListOfMovies**);  // Inserta una nueva película
int searchMovie(struct ListOfMovies*);  // Busca una película por su ID
int printAllMovies(struct ListOfMovies*);  // Imprime todas las películas
void cleanBuffer();  // Limpia el buffer

// Función principal
int main()
{
      struct ListOfMovies* head;  // Se crea un apuntador que será el comienzo de nuestra lista
      head = NULL;  // Se apunta hacia NULL al principio

      int option;

      // Descripción del programa
      printf("BASE DE DATOS DE PELÍCULAS\n");
      printf("\nEn esta base de datos el usuario puede almacenar los datos de cualquier cantidad de películas.
La base de datos está vacía ");
      printf("al comenzar el programa. Cada película tendrá un ID único, por lo que el usuario no debe ingresar
dos ID iguales. ");
      printf("El usuario debe elegir qué acción realizar dentro del siguiente menú.\n");

      do
      {
            // Menú de opciones
            printf("\nMenú de acciones:\n\t1. Registrar una nueva película en la base de datos\n");
            printf("\t2. Imprimir todas las películas almacenadas en la base de datos\n\t3. Buscar una
película en la base de datos\n\t");
            printf("4. Salir de la base de datos\n\n\tElija la acción que desea ejecutar: ");

            scanf("%d",&option);

            switch(option)
            {
                  case 1:
```

```
                        insertOneMovie(&head);  // Insertamos una pelicula tomando como argumento la
variable &head
                        break;
                case 2:
                        printAllMovies(head);  // Imprimimos todas las películas tomando como argumento la
variable head
                        break;
                case 3:
                        searchMovie(head);  // Buscamos una película por su ID
                        break;
                case 4:
                        return 0;  // Salimos del programa
                default:
                        printf("\nOpción inválida. Digite una de las cuatro opciones determinadas.\n");
        }
    } while(1);  //Un bucle infinito

    return 0;
}


void insertOneMovie(struct ListOfMovies** start)  // La función toma como argumento un apuntador doble
{
    printf("\n\nREGISTRO DE DATOS DE UNA NUEVA PELÍCULA\n");

    // Se crea un apuntador llamado temp que apunte a nuestra nueva estructura
    struct ListOfMovies* temp = (struct ListOfMovies*)calloc(1 ,sizeof(struct ListOfMovies));  // Se libera
espacio en memoria

    printf("\n\tIntroduzca un ID único para la película: ");
    scanf("%d", &(*temp).movieX.id);  // Se lee el ID

    struct ListOfMovies* count;
    count = *start;
    while(count != NULL)  // Se recorre la lista
    {
        if((*temp).movieX.id == (*count).movieX.id)  // Se verifica que no haya un ID repetido
        {
            printf("\nEl ID que ingresó ya se encuentra registrado en la base de datos. Inténtelo de
nuevo.\n");
            return;
        }
        count = (*count).next;
    }

    cleanBuffer();
    printf("\tIntroduzca el nombre de la película: ");
    fgets((*temp).movieX.name, 50, stdin);  // Se lee el nombre de la película

    // Se toman en cuenta los espacios dentro del nombre de la película
    if (((*temp).movieX.name > 0) && ((*temp).movieX.name[strlen((*temp).movieX.name) - 1] == '\n'))
   {
      (*temp).movieX.name[strlen((*temp).movieX.name) - 1] = '\0';
    }

   printf("\tIntroduzca el año de realización: ");
   scanf("%d", &(*temp).movieX.date);  // Se lee el año
   while( (*temp).movieX.date < 1900 || (*temp).movieX.date > 2021 )  // Se verifica que se ingrese un año
válido
   {
    printf("\tAño de realización inválido. Ingrese otro año: ");
    scanf("%d", &(*temp).movieX.date);
    }
```

```
    printf("\tIntroduzca el género: ");
    scanf("%s", &(*temp).movieX.genre);  // Se lee el género

    printf("\tIntroduzca la calificación (0.0-10.0): ");
    scanf("%f", &(*temp).movieX.score);  // Se lee la calificación
    while( (*temp).movieX.score < 0.0 || (*temp).movieX.score > 10.0 )  // Se verifica que se ingrese algo
válido
    {
      printf("\tCalificación inválida. Ingrese otra calificación: ");
      scanf("%f", &(*temp).movieX.score);
      }

      (*temp).next = *start;  // El apuntador next se apunta hacia donde apuntaba el head
      *start = temp;  // El head se apunta hacia donde apuntaba el temp

      printf("\t\t\t¡Los datos de la película fueron guardados exitosamente!\n");
}


int printAllMovies(struct ListOfMovies* count)
{
      printf("\n\nIMPRESIÓN DE LAS PELÍCULAS EN LA BASE DE DATOS\n");

      if(count == NULL)  // Si no hay datos, no se imprime nada
      {
             printf("\n\tADVERTENCIA: La base de datos se encuentra vacía. No hay películas para mostrar.\n");
             return 3;
      }

      printf("\nLa lista de todas las películas es:");

      while(count != NULL)  // Se recorre toda la lista
      {
             //Se imprime cada película y sus respectivos datos
             printf("\n\tID: %d", (*count).movieX.id);
             printf("\n\tName: %s", (*count).movieX.name);
             printf("\n\tDate: %d", (*count).movieX.date);
             printf("\n\tGenre: %s", (*count).movieX.genre);
             printf("\n\tScore: %.1f\n", (*count).movieX.score);
             count = (*count).next;  // Se itera la variable count
      }

      printf("\t\t\t¡Todas la películas fueron impresas exitosamente!\n");
      return 3;
}


int searchMovie(struct ListOfMovies* count)
{
      printf("\n\nBUSCADOR DE PELÍCULAS EN LA BASE DE DATOS\n");

      if(count == NULL)  // Si no hay datos, no se busca nada
      {
             printf("\n\tADVERTENCIA: No se pueden buscar películas debido a que la base de datos se encuentra
vacía.\n");
             return 3;
      }

      int ID;
      printf("\n\tIntroduzca el ID de la película que desea buscar en la base de datos: ");
      scanf("%d", &ID);  // Se lee el ID que se quiere buscar
```

```
	while(count != NULL)  // Se recorre la lists
	{
		if(ID == (*count).movieX.id)  // Si se encuentra un ID idéntico, se imprime el resultado
		{
			// Se imprimen los datos de la película
			printf("\nEncontramos la siguiente película con el ID ingresado:\n");
			printf("\n\tID: %d", (*count).movieX.id);
			printf("\n\tName: %s", (*count).movieX.name);
			printf("\n\tDate: %d", (*count).movieX.date);
			printf("\n\tGenre: %s", (*count).movieX.genre);
			printf("\n\tScore: %.1f\n", (*count).movieX.score);
			printf("\t\t\t¡La búsqueda ha finalizado exitosamente!\n");
			return 3;
		}

		count = (*count).next;// Se itera el count
	}

	printf("\n\t\tLo sentimos. No se encontró ninguna película con ese ID.\n");
	return 3;
}

void cleanBuffer()  // Función igual al fflush
{
	int ch;
	while((ch = fgetc(stdin)) != '\n') ;
}
```