Practica 1

Nota final	0

🔷 Estructura del programa

1. Variables globales

```
static int static_var = 0;   // Variable estática en el segmento de datos
int bss_var;                 // Variable global no inicializada (BSS)
```

- `static_var`: se guarda en el segmento de **datos** y conserva su valor aunque el programa avance.
- `bss_var`: al no estar inicializada, se guarda en el **segmento BSS**.

---

2. Enumeración de géneros de libros

```
typedef enum {
    FICTION, NON_FICTION, SCIENCE, HISTORY, FANTASY, BIOGRAPHY, OT
HER
} genre_t;
```

Representa categorías de libros (ficción, ciencia, historia, etc.).

---

3. Estructuras

- **Libro (`book_t`)**

```
typedef struct _book {
    int id;
    char title[100];
    char author[100];
    int publication_year;
    genre_t genre;
```

```
    int quantity;
    struct _book *next;   // Lista enlazada de libros
} book_t;
```

- **Miembro (`member_t`)**

```
typedef struct _member {
    int id;
    char name[100];
    int issued_count;     // Cuántos libros tiene prestados
    int *issued_books;    // Arreglo dinámico con IDs de libros prestados
    struct _member *next; // Lista enlazada de miembros
} member_t;
```

Ambos usan **listas enlazadas** para almacenar múltiples registros.

---

4. Funciones principales

🔶 Conversión de género

```
const char* genreToString(genre_t genre)
```

Convierte el valor del enum (`FICTION`, etc.) a texto legible ("Ficcion", "Historia", etc.).

🔶 Manejo de libros

- **Agregar libro** → `addBook()`
  - Pide datos al usuario (ID, título, autor, año, género, cantidad).
  - Reserva memoria con `malloc`.
  - Inserta el libro al inicio de la lista.
  - Incrementa el contador de libros.
- **Buscar libro por ID** → `findBookById()`

- **Mostrar libros** → `displayBooks()` usa `displayBooksRecursive()` para imprimir la lista.
- **Liberar memoria** → `freeLibrary()` libera todos los libros con `free`.
- **Guardar/Cargar desde archivo** → `saveLibraryToFile()`, `loadLibraryFromFile()`

🔶 Manejo de miembros

- **Agregar miembro** → `addMember()`
  - Similar a `addBook`, pero con campos de miembros.
  - `issued_books` inicia en `NULL`.
- **Prestar libro** → `issueBook()`
  - Disminuye la cantidad del libro.
  - Aumenta el contador de libros prestados del miembro.
  - Usa `realloc` para agrandar dinámicamente la lista `issued_books`.
- **Devolver libro** → `returnBook()`
  - Incrementa cantidad del libro.
  - Elimina el ID del arreglo `issued_books` del miembro (también con `realloc`).
- **Mostrar miembros** → `displayMembers()`
  - Muestra todos los miembros y sus libros prestados.
- **Buscar miembro** → `searchMember()`
  - Encuentra un miembro por ID y muestra sus datos.
- **Liberar memoria** → `freeMembers()`
  - Libera tanto el arreglo `issued_books` como la estructura `member`.
- **Guardar/Cargar desde archivo** → `saveMembersToFile()`, `loadMembersFromFile()`

---

5. `main()`

```
int main() {
    book_t *library = NULL;
    member_t *members = NULL;
    int bookCount = 0, memberCount = 0;
    int choice = 0;
```

```
    loadLibraryFromFile(&library, &bookCount, "library.txt");
    loadMembersFromFile(&members, &memberCount, "members.txt");

    do {
        // Menú con opciones
        switch (choice) {
            case 1: addBook(&library, &bookCount); break;
            case 2: displayBooks(library); break;
            case 3: addMember(&members, &memberCount); break;
            case 4: issueBook(library, members); break;
            case 5: returnBook(library, members); break;
            case 6: displayMembers(members, library); break;
            case 7: searchMember(members, library); break;
            case 8:
                saveLibraryToFile(library, "library.txt");
                saveMembersToFile(members, "members.txt");
                printf("Saliendo del programa\n");
                break;
            default:
                printf("Esta no es una opcion valida!!!\n");
        }
    } while(choice != 8);

    freeLibrary(library);
    freeMembers(members);
    return 0;
}
```

El menú permite al usuario interactuar con el sistema de biblioteca.

🔷 Puntos clave del código

1. **Gestión dinámica de memoria** con `malloc`, `realloc`, `free`.
2. **Uso de listas enlazadas** para manejar una biblioteca y sus miembros.
3. **Persistencia de datos** en archivos (`.txt`) para no perder la información al cerrar el programa.

4. **Segmentos de memoria** demostrados:
    - `static_var` → segmento de **datos**.
    - `bss_var` → segmento **BSS**.
    - `library`, `members`, `bookCount`, etc. → **stack**.
    - `malloc/realloc` → **heap**.