Universidad Continental

Sistema de gestión de Biblioteca

Integrantes:Juan Carlos Llactahuamani
Cristian Josue Mullisaca Guzman
Paolo Jesus Medina Churana

NRC: 28491

Problemática:

Una biblioteca online enfrenta dificultades para administrar eficientemente su catálogo de libros, gestionar las reservas y devoluciones, y facilitar búsquedas rápidas, lo que ocasiona desorganización y retrasos en la atención a los usuarios.

DISEÑO DE SOLUCIÓN:

Implementar un sistema de gestión que permita organizar el catálogo, manejar reservas en cola, registrar devoluciones mediante una pila, y realizar búsquedas y ordenamientos eficientes, mejorando así la administración y experiencia tanto para el personal como para los usuarios.

Solución implementada:

```
1   #include <iostream>
2   #include <string>
3   using namespace std;
4
5   struct Libro {
6       string titulo;
7       Libro* siguiente;
8   };
```

#include <iostream>: Permite usar cin y cout para **entrada/salida.**

#include <string>: Permite usar cadenas de texto (string).

using namespace std;: Nos evita tener que escribir std:: antes de cosas como cout.

Define un libro que se guarda en una lista enlazada.

Tiene un título y un puntero al siguiente libro.

Representa una reserva en una cola.

Guarda el nombre del estudiante y el título del libro.

Representa una devolución en una pila.

Guarda el título del libro devuelto.

```
struct Libro {                  /.
    string titulo;              /.
    Libro* siguiente;           /.
};

struct NodoCola {
    string estudiante, libro;
    NodoCola* sig;
};

struct NodoPila {
    string libro;
    NodoPila* sig;
};
```

Funciones para manejar libros

```
void agregar(Libro*& cabeza, string t) {
    Libro* nuevo = new Libro{t, NULL};
    if (!cabeza)
        cabeza = nuevo;
    else {
        Libro* temp = cabeza;
        while (temp->siguiente)
            temp = temp->siguiente;
        temp->siguiente = nuevo;
    }
}
```

Agrega un nuevo libro con el título t al **final de la lista enlazada**
Se crea un nodo llamado nuevo.

Su campo título guarda el texto ingresado.

Su campo siguiente apunta a NULL porque será el **último** en la lista.

Si la lista está vacía (cabeza == NULL), el nuevo nodo se convierte en el primero (la cabeza).Si ya hay libros, se recorre la lista con temp hasta encontrar el último nodo.
Luego se **conecta** el nuevo libro al final de la lista.

Muestra todos los libros en el catálogo, **uno por uno**

Usa un puntero temp que empieza en la cabeza de la lista.

Recorre cada nodo mientras no sea NULL.

Imprime el título del libro en cada nodo.

```
void mostrar(Libro* cabeza) {
    for (Libro* temp = cabeza; temp; temp = temp->siguiente)
        cout << "- " << temp->titulo << "\n";
}
```

```
void ordenar(Libro* cabeza) {
    if (!cabeza) return;
    bool cambiado;
    do {
        cambiado = false;
        for (Libro* a = cabeza; a && a->siguiente; a = a->siguiente) {
            if (a->titulo > a->siguiente->titulo) {
                swap(a->titulo, a->siguiente->titulo);
                cambiado = true;
            }
        }
    } while (cambiado);                    // Repetir mientras haya cambios
}
```

Ordena los libros del catálogo alfabéticamente por título, usando el algoritmo de burbuja.
Si la lista no tiene libros, no hace nada
Declara un booleano cambiado para saber si hubo intercambios en esta pasada
Compara cada título con el siguiente:

Si a->titulo está después alfabéticamente que a->siguiente->titulo, entonces están desordenados.

Usa swap para intercambiar los títulos.Marca cambiado = true para indicar que se hicieron cambios y se necesita otra pasada

El ciclo se repite hasta que en una pasada no se realice ningún intercambio.

Eso significa que la lista está ordenada.

```
void encolar(NodoCola*& frente, NodoCola*& fin, string e, string l) {
    NodoCola* nuevo = new NodoCola{e, l, NULL};
    if (fin)
        fin->sig = nuevo;
    else
        frente = nuevo;
    fin = nuevo;
}

void mostrarReservas(NodoCola* frente) {
    for (NodoCola* t = frente; t; t = t->sig)
        cout << "- " << t->estudiante << " reservó: " << t->libro << "\n";
}
```

Crea un nuevo nodo llamado `nuevo`:

Guarda el nombre del estudiante `e`.
Guarda el nombre del libro `l`.
`sig = NULL` porque será el último nodo en la cola.Si la cola no está vacía (`fin` apunta al último elemento), conecta el último nodo actual con el nuevo nodo (`fin->sig = nuevo`).Si la cola está vacía, el nuevo nodo se convierte en el primer elemento (`frente`).Ahora el nuevo nodo es también el último de la cola, así que actualizamos el puntero `fin`.Recorre la cola desde el primer nodo (`frente`) hasta el final (`NULL`):

Se usa un puntero temporal `t`. Muestra en pantalla el nombre del estudiante y el libro que reservó.

Crea un nuevo nodo con el libro `l` y lo **coloca en la cima de la pila**.

`l`: es el nombre del libro devuelto.
`cima`: apunta al nodo anterior (el que estaba en la cima antes).
Ahora, el nuevo nodo es el **primer elemento de la pila**.

```
void apilar(NodoPila*& cima, string l) {
    cima = new NodoPila{l, cima};
}

void mostrarDevoluciones(NodoPila* cima) {
    for (NodoPila* t = cima; t; t = t->sig)
        cout << "- " << t->libro << "\n";
}
```

```
void mostrarDevoluciones(NodoPila* cima) {
    for (NodoPila* t = cima; t; t = t->sig)
        cout << "- " << t->libro << "\n";
}
```

Recorre la pila desde la cima hacia abajo (hasta llegar a NULL),muestra el título del libro que está en ese nodo.