Universidad Central Marta Abreu de las Villas.

Facultad: Matemática, Física y Computación.

Título Proyecto: Biblioteca.

Autor: Carlos Miguel Vázquez Corona

**Problema**

El problema consistía en diseñar un sistema de biblioteca en C++ que permitiera gestionar libros, incluyendo funcionalidades como agregar libros, prestar y devolver libros, mostrar información de libros disponibles y prestados, entre otras.

Para resolver este problema, se siguieron los siguientes pasos:

**Diseño de Clases:**

+-------------------------------------+

| Biblioteca |

+-------------------------------------+

| - libros: vector<Libro\*> |

| - revistas: vector<Revista\*> |

+-------------------------------------+

| + agregarLibro(libro: Libro) |

| + agregarRevista(revista: Revista) |

| + eliminarLibro(titulo: string) |

| + eliminarRevista(titulo: string) |

| + mostrarLibros() |

| + mostrarRevistas() |

| + prestarLibro(titulo: string, |

| persona: Persona\*) |

| + devolverLibro(titulo: string) |

| + devolverRevista(titulo: string) |

| + mostrarLibrosPrestados() |

+-------------------------------------+

+-------------------------------------+

| Libro |

+-------------------------------------+

| - titulo: string |

| - autor: string |

| - prestado: bool |

+-------------------------------------+

| + Libro(titulo: string, autor: |

| string) |

| + getTitulo(): string |

| + getAutor(): string |

| + isPrestado(): bool |

| + setPrestado(prestado: bool) |

+-------------------------------------+

+-------------------------------------+

| Revista |

+-------------------------------------+

| - titulo: string |

| - saga: string |

| - estante: string |

| - prestado: bool |

+-------------------------------------+

| + Revista(titulo: string, saga: |

| string, estante: string) |

| + getTitulo(): string |

| + getSaga(): string |

| + getEstante(): string |

| + isPrestado(): bool |

| + setPrestado(prestado: bool) |

+-------------------------------------+

+-------------------------------------+

| Persona |

+-------------------------------------+

| - nombre: string |

| - apellido: string |

+-------------------------------------+

| + Persona(nombre: string, |

| apellido: string) |

| + getNombre(): string |

| + getApellido(): string |

+-------------------------------------+

Se diseñaron las clases Libro, Persona y Biblioteca. La clase Libro representa un libro con sus atributos como título, autor, ISBN, páginas, edición, editorial y estado de préstamo. La clase Persona representa a una persona con su nombre y apellido. La clase Biblioteca gestiona una colección de libros y proporciona métodos para agregar libros, prestar y devolver libros, y mostrar información de libros disponibles y prestados.

**Implementación de Funcionalidades:**

Se implementaron las funcionalidades requeridas, como agregar y eliminar libros, prestar y devolver libros, y mostrar información de libros disponibles y prestados. Además, se incluyó un mecanismo de administrador para gestionar la biblioteca.

**Interfaz de Usuario:**

Se creó un menú interactivo que permite al usuario elegir entre diferentes opciones, como mostrar libros, prestar un libro, devolver un libro, agregar un libro (solo para administradores), eliminar un libro (solo para administradores), cambiar entre modo administrador y modo usuario, y salir del programa.

**Casos de Uso**

Agregar libro o revista:

Actor: Bibliotecario/Administrador.

Descripción: El bibliotecario añade un nuevo libro o revista al sistema, proporcionando título, autor (en el caso de libros) y otros detalles pertinentes.

Eliminar libro o revista:

Actor: Bibliotecario/Administrador.

Descripción: El bibliotecario elimina un libro o revista del sistema basado en su título. Esto puede ser útil para retirar elementos obsoletos o incorrectos.

Mostrar libros o revistas disponibles:

Actor: Usuario/Bibliotecario.

Descripción: Los usuarios pueden ver una lista de todos los libros o revistas disponibles en la biblioteca en un momento dado. Esto les ayuda a conocer qué recursos pueden utilizar.

Prestar libro o revista:

Actor: Bibliotecario/Administrador.

Descripción: El bibliotecario presta un libro o revista a un usuario registrado en la biblioteca. Esto implica actualizar el estado del recurso a "prestado" y registrar los detalles del préstamo, como la fecha de vencimiento.

Devolver libro o revista:

Actor: Usuario/Bibliotecario.

Descripción: El usuario devuelve un libro o revista prestado a la biblioteca. Esto implica actualizar el estado del recurso a "disponible" y registrar los detalles del retorno, como la fecha de devolución.

Mostrar libros o revistas prestados:

Actor: Bibliotecario/Administrador.

Descripción: El bibliotecario puede ver una lista de todos los libros o revistas actualmente prestados, junto con la información del usuario que los ha prestado y las fechas de vencimiento.

Cambiar a modo administrador o cliente:

Actor: Usuario.

Descripción: Los usuarios pueden cambiar entre el modo de cliente y el modo de administrador según sus privilegios. En el modo de administrador, tienen acceso a funcionalidades adicionales como agregar o eliminar libros y revistas.

**Beneficios**

Gestión Eficiente: El sistema permite una gestión eficiente de la biblioteca, lo que facilita a los usuarios encontrar y acceder a los libros disponibles.

Mejora la Experiencia del Usuario: La interfaz de usuario intuitiva y amigable hace que sea fácil de usar para todos los usuarios, tanto administradores como clientes.

Control de Préstamos: El sistema controla los préstamos de libros, lo que garantiza que los libros prestados sean devueltos a tiempo y se mantenga un registro de quién tiene cada libro en préstamo.

Uso de C++: El desarrollo del sistema en C++ ofrece varios beneficios, como rendimiento eficiente, portabilidad y flexibilidad. Además, el uso de características avanzadas de C++ como la programación orientada a objetos mejora la modularidad y la reutilización del código.

**Importancia de la Orientación a Objetos y C++**

En este proyecto, la programación orientada a objetos (POO) desempeña un papel fundamental. La POO permite modelar el sistema de biblioteca de manera más natural al representar los conceptos del mundo real, como libros y personas, como objetos con sus propias propiedades y comportamientos.

**Los beneficios de la orientación a objetos en este caso incluyen:**

Modelado Realista: La POO permite modelar entidades del mundo real como clases y objetos, lo que facilita la comprensión y el diseño del sistema.

Modularidad y Reutilización: La POO promueve la modularidad al dividir el sistema en clases cohesivas y reutilizables. Esto facilita la incorporación de nuevas funcionalidades y el mantenimiento del código.

Encapsulamiento: La encapsulación permite ocultar los detalles internos de una clase y exponer solo la interfaz pública. Esto facilita la gestión de la complejidad y mejora la seguridad del sistema.

Herencia y Polimorfismo: La herencia y el polimorfismo permiten extender y especializar el comportamiento de las clases. En este caso, se podría extender la clase Libro para crear subclases como Comic, Periodico y Revista, que heredan propiedades y comportamientos comunes de la clase base Libro pero pueden tener comportamientos específicos adicionales.

El uso de C++ como lenguaje de programación ofrece aún más ventajas, como la eficiencia de ejecución, la capacidad de control de bajo nivel y la compatibilidad con una amplia gama de plataformas. Además, las características avanzadas de C++, como la programación orientada a objetos, los punteros y las plantillas, brindan flexibilidad y potencia para implementar soluciones complejas de manera efectiva.

**Conclusiones**

En conclusión, el desarrollo de este sistema de biblioteca en C++ aprovecha las ventajas de la orientación a objetos y el poder de C++ como lenguaje de programación. La implementación eficiente de funcionalidades clave, junto con una interfaz de usuario intuitiva, mejora la experiencia del usuario y facilita la gestión de la biblioteca. Además, el diseño modular y orientado a objetos del sistema permite una fácil extensibilidad y mantenimiento a largo plazo.

**Enseñanzas Obtenidas**

Este proyecto proporcionó una valiosa experiencia en el diseño e implementación de sistemas de gestión de información utilizando principios de programación orientada a objetos y características avanzadas de C++. Aprendimos sobre la importancia