Práctica 1: Implementación de una biblioteca

Fecha de entrega: 21 de Noviembre
Fecha de la defensa: 22 de Noviembre

Objetivo: Primera aproximación a C++: estructuras de control, tipos e iniciación a la orientación a objetos.

Material proporcionado:

Fichero	Explicación
`tests.zip`	Proyecto de tests para pruebas de unidad
`validador.zip`	Programa validador de la entrega.
`documentacion.zip`	Documentación de ficheros y clases necesarias para la implementación.

Esta primera práctica consiste en la creación de una pequeña aplicación para gestionar una biblioteca, incluyendo la lista de libros, la base de datos de los socios y una pequeña gestión de préstamos.

La práctica consta de dos partes que se traducirán en sendos proyectos. La diferencia entre ambos será simplemente de implementación: la primera parte utilizará tipos tradicionales de C (y también Pascal), como son enumerados y estructuras (registros), mientras que la segunda parte utilizará clases y objetos.

1. Descripción de la biblioteca

Esta sección describe las capacidades de gestión que tendrá la aplicación.

1.1. Menú principal

Al principio de la aplicación se le presentará al usuario un menú con tres opciones:

```
1 - Gestion de libros
2 - Gestion de socios
3 - Gestion de la biblioteca
0 - Salir
```

La primera opción permite añadir y eliminar libros, la segunda sirve para añadir y eliminar socios, y la última opción se utiliza para la gestión de los préstamos.

Cada una de las opciones son descritas con detalle a continuación.

1.2. Gestión de la base de datos de libros

Al entrar en la gestión de libros se presenta un nuevo menú con 4 posibilidades (más la de salir):

```
1 - Añadir un nuevo libro
2 - Eliminar un libro
3 - Listar todos los libros
4 - Mostrar los datos de un libro
0 - Volver al menu principal
```

La primera opción pregunta al usuario los datos de un libro (título y autor), así como si el libro estará disponible para préstamo o es de uso exclusivo en la sala. Tras estas preguntas, la biblioteca asignará un número de registro único a ese libro y lo añadirá a la biblioteca. El número de registro del primer libro añadido será el 1 y se irá incrementando con la llegada de cada nuevo libro.

La segunda opción permite al usuario eliminar un libro previamente registrado. La aplicación preguntará el número de registro del libro, y éste se eliminará *únicamente si el libro no está prestado.*

La tercera opción presenta una lista exhaustiva de todos los libros que hay en la biblioteca. Para cada libro se mostrará el número de registro, título, autor y su estado: disponible, prestado o no disponible[1]. Por ejemplo:

```
1: Tuareg (Alberto Vazquez Figueroa) [D]
2: Groucho y yo (Groucho Marx) [P]
3: Longitud (Dava Sobel) [ND]
```

[1] Un libro se considera *no disponible* cuando es un libro para uso exclusivo en sala.

El primer libro está disponible (`[D]`) el segundo está prestado (`[P]`) y el tercero sólo puede consultarse en sala, por lo que no está disponible (`[ND]`).

Por último, con la cuarta opción el usuario podrá consultar los datos de un único libro a partir del número de registro. La aplicación mostrará por pantalla la información del libro en el mismo formato que la opción anterior.

1.3. Gestión de la base de datos de socios

Igual que la anterior, la gestión de socios presenta un menú adicional, con opciones similares a las del menú de gestión de libros:

```
1.- Añadir socio
2.- Eliminar socio
3.- Listar socios
4.- Mostrar datos de un socio
0.- Volver
```

Para añadir un socio se le solicitará al usuario el nombre y los apellidos del socio, así como el tipo de socio que es (Alumno, Becario o Profesor). Tras esto, la biblioteca asignará un código de socio único y lo añadirá a la base de datos. El primer socio tendrá código 101, y se irá incrementando con la llegada de los nuevos socios.

Para eliminar un socio el usuario deberá indicar el código del socio y la biblioteca lo eliminará *únicamente si no tiene libros prestados*.

Tanto la opción 3 como la opción 4 consisten en mostrar la información de uno o varios socios. Para cada socio se indicará su código, nombre y apellidos, el tipo de socio que es y a continuación la información de los libros que tiene prestados. Por ejemplo:

```
101: Walterio Malatesta (profesor)
Sin libros prestados
102: Marty McFly (alumno)
2: Groucho y yo (Groucho Marx) [P]
```

1.4. Gestión de préstamos

La última opción del menú principal conduce a la gestión de préstamos:

```
1 - Registrar un nuevo prestamo
2 - Devolucion de un libro
3 - Listado de prestamos
4 - Buscar un libro
0 - Volver al menu principal
```

La primera opción registra un nuevo préstamo dado el código del socio, el número de registro del libro y la fecha de la devolución. Hay que tener en cuenta que para poder realizar un préstamo, el libro debe estar disponible y el socio no debe haber superado el número máximo de libros prestados. Ese máximo depende del tipo de socio: los profesores pueden coger hasta 8 libros, los becarios 5 y los alumnos 3.

La segunda opción sirve para devolver un libro. Igual que antes, la aplicación preguntará tanto por el código del socio que devuelve el libro como por el número de registro de éste.

El listado de préstamos presenta por pantalla una lista de todos los libros que hay prestados así como el nombre y apellidos del socio que lo tiene. Por ejemplo:

```
2: Groucho y yo (Groucho Marx) [P] => Prestado a Marty McFly
```

La última opción del menú permite averiguar quién tiene un libro, dado su número de registro. Si el libro está prestado aparecerá un mensaje del tipo

```
El libro lo tiene Marty McFly.
```

y si no está prestado se indicará

```
Ese libro no lo tiene nadie.
```

En las secciones siguientes aparecen detalles que deben tenerse en cuenta a la hora de implementar la práctica.

2. Implementación de la biblioteca sin orientación a objetos

Para la implementación necesitarás implementar los módulos que aparecen a continuación[2]. En la documentación que acompaña a este enunciado (fichero `documentacion.zip`) aparecen descritos con más detalle los elementos que deben contener[3]:

- `Fecha`: define el tipo `TFecha` que se utiliza para guardar las fechas de devolución de los préstamos.
- `Libro`: contiene la definición del enumerado (`TEstado`) que indica los posibles estados en los que puede estar un libro (disponible, prestado o no disponible), así como la estructura y funciones para trabajar con libros (`TLibro`).

[2]Entendemos por módulo la pareja de ficheros con extensión `.h` y `.cpp` con el nombre indicado.

[3]El orden de aparición no es arbitrario; aconsejamos seguir este orden para la implementación.

- `ListaLibros`: con el tipo `TListaLibros` y las funciones para trabajar con ella. Tanto en este caso como en el resto de las listas que utilizamos en esta práctica, están implementadas utilizando una estructura con un vector y un contador. En este caso el número máximo de libros almacenados será la constante `MAX_LIBROS` que deberá definirse como mínimo a 5.

- `Prestamo`: con el tipo `TInfoPrestamo` que guarda la información asociada a un único préstamo, a saber, el número de registro del libro prestado y la fecha de devolución.

- `ListaPrestamos`: con el tipo `TListaPrestamos` cuyo número máximo de elementos será `MAX_ELEMENTOS`.

- `Socio`: contiene la definición del enumerado `TTipoSocio` que puede tener los tres valores mencionados anteriormente (Profesor, Becario o Alumno). También tiene el tipo `TSocio`, que guarda el nombre, apellidos y tipo de socio, así como el código de socio y la lista de préstamos.

- `ListaSocios`: con el tipo `TListaSocios` cuyo número máximo de elementos será `MAX_SOCIOS`.

- `Biblioteca`: que define el tipo `TBiblioteca` y las funciones útiles para su gestión.

El punto de entrada al programa estará en el fichero `main.cpp`. La función `main` tendrá una variable local del tipo `TBiblioteca` que será la que se utilice durante todo el programa.

3. Implementación de la biblioteca orientada a objetos

El funcionamiento de la segunda parte de la práctica debe ser *exactamente el mismo* que el de la primera parte. La única diferencia debe ser la implementación.

Para realizarla, se creará un proyecto nuevo (integrado en la misma solución), cuyo punto de entrada aparecerá en el fichero `mainOO.cpp`. Existirán los mismos módulos anteriores pero añadiendo a sus nombres el sufijo `OO`[4]. Todas las estructuras definidas pasarán a ser clases cuyo nombre será el mismo que la estructura pero comenzando por la letra `C` en lugar de la `T`. Por último, los dos tipos enumerados definidos mantendrán su nombre, pero pasarán a ser tipos *internos* definidos en las clases `CLibro` y `CSocio`.

[4] El `OO` nos indica que son "orientados a objetos" para distinguirlos de los módulos de la primera parte.

Todas las funciones implementadas en la primera parte pasarán a ser *métodos* de las clases definidas, mientras que los campos de los registros pasarán a ser *atributos protegidos*[5]. Por su parte, las funciones de inicialización pasarán a ser los constructores de las clases. Se deberá tener especial cuidado en la traducción de los parámetros constantes, así como en la conversión de los métodos que *no* recibían como parámetro la estructura del módulo al que pertenecían.

Debido al proceso de la traducción y al hecho de que los atributos de las clases pasan a ser protegidos, te verás en la obligación de añadir métodos para acceder y alterar los campos. En esta primera práctica utilizaremos para eso métodos del estilo `dameNumRegistro` para acceder al campo `_numRegistro` de la clase `CLibro` o `ponCodigo(int)` para cambiar el código de un `CSocio`.

Por último, ten presente que la definición de las constantes que indican el tamaño máximo de listas debe utilizar la forma recomendada al programar en C++: el uso de `enum` con inicialización explícita de los valores:

```
enum { MAX_LIBROS = 50 };
```

Esta definición de constantes es preferible pues, entre otras cosas, se puede añadir a la parte pública de la clase, minimizando la posbilidad de colisión de nombres.

4. Tests

Para esta práctica se proporciona un proyecto de test que incluye las pruebas de las dos partes. Como en todas las prácticas del curso, es *obligatorio* que la práctica entregada supere esos tests.

Como se puede comprobar en el proyecto, existe un fichero de test *por cada módulo*. Eso permite pasar los test de manera incremental según se van implementando los distintos módulos. Para eso, basta con ir eliminando temporalmente del proyecto aquellos ficheros de test que pongan a prueba los módulos aún no implementados.

5. Normas de entrega

Es indispensable que el código fuente supere los tests de unidad proporcionados y que el fichero enviado pase el programa validador publicado.

[5] Como nomenclatura, aconsejamos que sus nombres comiencen por el guión bajo, "_".