Estructuras de Datos y de la Información

1ª Entrega. Curso 2021/22

Gestión de una Biblioteca

Por:
Profesores de EDI

Índice general

1. Introducción

Este documento describe los requerimientos y funcionalidad básica para el desarrollo de la primera entrega de programación de la asignatura *Estructuras de Datos y de la Información* del curso 2021-22.

En esta entrega vamos a implementar una aplicación para gestionar una *Biblioteca*, en la que tendremos un catálogo de libros y una serie de usuarios a los que podemos prestar libros.

Para el desarrollo de la aplicación conforme a los requisitos de la asignatura se seguirá una metodología de desarrollo basada en el paradigma de *Programación Orientada a Objetos* y se utilizará el lenguaje de programación *C++*.

Este documento se organiza de la siguiente forma. En la sección 2 se discuten aspectos generales del desarrollo de la aplicación, incluidas sus clases y estructuras de datos. En la sección 3 se enumeran los algoritmos básicos que debemos implementar. Finalmente, la sección 4 establece los **criterios de evaluación** y calificación, así como las **fechas de entrega**.

2. Metodología e Implementación

En esta sección se describe la funcionalidad que implementará nuestra aplicación, y se establecen una serie de pasos generales que pueden servir de guía para el desarrollo de la misma.

En el desarrollo partirá de una plantilla de código similar a la que venimos utilizando en las clases de laboratorio, y que está disponible en el aula virtual de la asignatura. La plantilla incluye los ficheros de datos necesarios.

La aplicación comenzará mostrando un menú en pantalla con tantas opciones como algoritmos a implementar (descritos en la seccion 3).

2.1. Ficheros de Datos

Los datos con los que trabajaremos se proporcionan en dos ficheros[1] de texto. Estos ficheros se describen en la Tabla 1. Los campos de cada archivo son autoexplica-

[1]Se deben utilizar los datos en los ficheros proporcionados sin modificaciones. Cualquier modificación supondrá que los resultados de los algoritmos no serán los esperados, y como consecuencia se podrán considerar incorrectos.

tivos, y es labor del programador extraer el tipo de cada dato para representarlo adecuadamente.

Nombre	**Descripción**
`libros.csv`	Fichero que contiene los *libros* disponibles en la biblioteca.
`usuarios.csv`	Fichero que contiene los *usuarios* de la biblioteca.

Tabla 1: Ficheros de datos para la aplicación.

Los ficheros se cargarán en las estructuras de datos de la aplicación al **inicio de la ejecución** (antes de mostrar el menú por pantalla). Al finalizar la aplicación, se almacenarán los datos de forma que se puedan volver a cargar en la siguiente ejecución.

2.2. Implementación de las clases *base*

La clases básicas a implementar en la aplicación serán las que representan un `Libro` y un `Usuario`. Deben contener los atributos (privados) necesarios para implementar la funcionalidad de nuestra aplicación, y las operaciones para el acceso a los atributos. Las operaciones incluirán constructores y destructores, métodos modificadores y selectores (es decir, *poner* y *obtener*), y algunos otros necesarios como el método para *mostrar* por pantalla la información.

Los usuarios deberán contener un campo que indique el libro que tienen prestado (en caso de no tener ningún libro prestado se pondrá a un valor por defecto). Este valor será almacenado en el fichero, de forma que se pueda mantener esta información entre ejecuciones.

2.3. Contenedores de datos

La estructura de datos básica que vamos a utilizar en nuestra aplicación para almacenar la información y procesarla es el *Vector de Ocupación Variable* (VoV). La aplicación de gestión de la Biblioteca, por tanto, contendrá un VoV para almacenar la información de los usuarios, y otro para almacenar la información de los libros. Los VoV no estarán ordenados, con lo que se simplifica la implementación de sus métodos para insertar y borrar elementos.

Los vectores de ocupación variable tendrán las siguientes operaciones[2]:

- **insertar(libro)**: inserta un libro en el VoV y aumenta en 1 su ocupación.
- **eliminar(id)**: elimina un libro por su identificador.
- **obtener(pos, ↑libro)**: obtiene un libro dada su posición en el vector.
- **obtener(id, ↑libro)**: obtiene un libro por su identificador.
- **existe(id): bool**: deveulve `true` si un libro existe en el VoV.
- **cuantos(): int**: devuelve el número de libros en el vector.

Esta interfaz se considera suficiente para el tratamiento de los *libros* del vector. En el caso de *usuarios*, la interfaz será muy similar. Los algoritmos de nuestra aplicación especificados en la sección 3, se implementarán en la clase `Biblioteca`.

Los Vectores de Ocupación Variable tendrán una definición simlar a la siguiente:

```
1  class VoV_Libros {
2
3  private:
4      Libro* vov_libros [MAX_LIBROS];
5      int    ocupacion;
6
7  public:
8
9      // PRE  = {  }
10     // POST = { VoV_Libros.ocupacion = 0 }
11     // DESC:  Crea un VoV que contiene punteros a libros.
12     // COMPL: O(1)
13     VoV_Libros ();
14
15     // PRE  = {  }
16     // POST = {  }
17     // DESC:  Destruir el VoV de punteros a libros (no los
           libros)
18     // COMPL: O(1)
19     ~VoV_Libros ();
20
21     . . .
22 };
```

[2]Los vectores de ocupación variable a implementar para contener libros y usuarios son muy similares, y se puede reutilizar gran parte del código. Se muestra en la lista la interfaz para el VoV de libros.

En cualquier caso, los vectores almacenarán la información como **punteros a objetos**. No se aceptará ninguna implementación que no cumpla con este requisito.

2.4. Implementación de las clase *principal*

Finalmente, tendremos una clase que represente la `Biblioteca`, y que contendrá su nombre, y los vectores anteriores. **Esta clase implementará los algoritmos descritos en la sección 3 como métodos de la clase.** Tendrá la siguiente forma:

```
class Biblioteca {

private:
   string        nombre;
   VoV_Libros   *libros;
   VoV_Usuarios *usuarios;

public:
   // Pre  = { }
   // Post = { }
   // Desc:
   // Compl: O( )
   Biblioteca (string nombre);
    . . .
};
```

3. Algoritmos

Los algoritmos a implementar en nuestra aplicación se deben mostrar en un menú inicial. Antes de mostrar el menú, se deben *cargar los datos de la aplicación* en las estructuras de datos (VoVs) desde los ficheros que almacenan los libros y usuarios.

Después se mostrará el menú con las siguientes opciones:

1. Prestar un libro a un usuario (utilizando la referencia del libro y el DNI del usuario). No se podrá prestar un libro que no existe.
2. Mostrar los libros de un temática determinada (por subcadena).

3. Buscar un libro por su referencia (identificador).
4. Mostrar todos los libros.
5. Mostrar todos los usuarios.
6. Mostrar todos los usuarios que tienen algún libro prestado.
7. Descatalogar un libro. Esto es, eliminarlo de la biblioteca. Solamente se podrá eliminar si no se ha prestado.

0 Finalizar la aplicación. Se almacenarán los datos actuales en los archivos de usuarios y libros.

Todos los algoritmos deben ser implementados sobre las estructuras de datos en memoria, es decir, no están permitidas las implementaciones de los algoritmos directamente sobre los ficheros de datos.

3.1. Documentación

Todos los algoritmos y los métodos de los VoVs y Biblioteca deben estar correctamente documentados[3]. La **documentación** incluirá, al menos, para cada método las *Precondiciones* y *Postcondiciones*[4], complejidad (utilizando notación *Big-O*) y descripción de la funcionalidad del método u operación. Además, cada fichero de código contendrá en la parte superior, el nombre del autor y correo electrónico, fecha de última modificación, y descripción del contenido.

4. Criterios de Evaluación y fecha de entrega

La evaluación de la implementación tomará en cuenta los criterios generales que se especifican a continuación:

1. Uso de **punteros** en la implementación, tanto en variables como en estructuras de datos, evitando así los problemas derivados de mover datos en memoria (*eficiencia*) y mantener distintas copias (*consistencia*).

[3]No se requiere, dada su simplicidad, documentar los métodos dde las clases Libro y Usuario.

[4]Recordad que las Precondiciones y Postcondiciones son un conjunto de expresiones lógicas, y no texto descriptivo, y que una vez establecidas deben ser respetadas. Esto es, el programador que utiliza la interfaz debe cumplir las precondiciones, y el programador que implementa la interfaz debe asegurar las postcondiciones.

2. Organización correcta de los datos en memoria de forma que la implementación de los algoritmos cumpla con los requisitos de eficiencia y uso de memoria. Por ejemplo, solamente habrá una copia de cada libro y usuario en la aplicación.
3. Corrección en el resultado de la ejecución de los algoritmos. No se evaluará ningún código que no **compile correctamente y ejecute**.
4. El lenguaje de programación será C++. El entorno (IDE) y el sistema de operativo no se tendrá en cuenta, puesto que se entregán únicamente el código fuente desarrollado. No está permitido utilizar la STL[5] en el desarrollo, ni cualquier otra biblioteca similar.
5. Uso correcto y eficiente de parámetros en la invocación de operaciones.
6. Uso de constructores y destructores para las clases de forma coherente y adecuada, incluido el constructor por copia.
7. Siempre que sea posible, no se duplicarán objetos en la aplicación, y se evitarán copias y duplicados de los mismos. El uso de punteros nos ofrece esta capacidad.

Además, las siguientes normas se deben seguir en la realización del proyecto:

- La implementación y entrega es **individual**.
- La entrega consistirá en un fichero `zip` cuyo nombre seguirá el formato `Apellidos_Nombre.zip`, que contendrá el código fuente implementado y los archivos `.csv` (no será un proyecto Eclipse ni de cualquier otro IDE).
- Cualquier intento de copia, detección de entrega de un código que no es propio o desarrollado por terceros, será motivo de suspenso (0) en la asignatura (para todos los implicados), y puesta en conocimiento de la autoridad del centro.

La fecha límite de entrega del proyecto es el **martes, 1 de marzo**, a través de la correspondiente tarea abierta en el campus virtual de la asignatura. No se tendrán en cuenta entregas realizadas por cualquier otro medio que no sea el establecido, ni con posterioridad a esa fecha.

4.1. Criterios de Evaluación de Laboratorio

Tened en cuenta que la nota de laboratorio será **media de las entregas** del curso, siempre que se consiga **más de un** 3 **en cada entrega** (consultad la Ficha 12a para más información). Si no se realiza alguna entrega o se supende con una nota inferior a 3, se podrá recuperar en la **convocatoria extraordinaria** de la asignatura.

[5]Standard Template Library