PRÁCTICA Nº 2: 1 sesión (14, 15 y 16 de Marzo y 7 y 8 de Abril)

GESTIÓN DE UN LISTADO DE LIBROS (Versión orientada a objetos)

0. OBJETIVOS

a. Reconstrucción, mediante la utilización de clases y objetos, de un programa construido con una aproximación diferente.
b. Compilación separada.

1. REALIZACIÓN DE LA PRÁCTICA

Se pide reescribir, desde el punto de vista de la programación orientada a objetos, el programa realizado en la práctica 1 para la gestión de libros. El funcionamiento esperado para el nuevo programa es el mismo que tenía el programa de la práctica anterior. Lo que debe cambiar es la estructura y organización del código mediante la introducción de clases y la correspondiente manipulación de objetos.

El programa debe incluir 2 tipos de objetos (clases), uno que represente el concepto de `Libro` y otro que represente el concepto de `Biblioteca`.

La clase `Libro` debe permitir (mediante métodos públicos) la posibilidad de que se realicen las siguientes operaciones sobre un libro:

1) Asignar/modificar el autor del libro (`void CambiarAutor (string a);`)
2) Asignar/modificar el título del libro (`void CambiarTitulo (string t);`)
3) Conocer el autor del libro (`string Autor ();`)
4) Conocer el título del libro (`string Titulo ();`)
5) Mostrar por pantalla los datos del libro (`void MostrarLibro ();`)

La clase `Biblioteca` debe permitir (mediante métodos públicos) la posibilidad de que se realicen las siguientes operaciones sobre una biblioteca:

1) Cargar desde un archivo la información de los libros contenidos en la biblioteca (`bool LeerBiblioteca (string arch);`)
2) Mostrar por pantalla los datos de los libros almacenados en la biblioteca (`void MostrarBiblioteca ();`)
3) Ordenar por autor los libros de la biblioteca (`void OrdenarBiblioteca ();`)
4) Buscar todos los libros de un autor almacenados en la biblioteca (`void BuscarPorAutor (string a);`)
5) Buscar un libro en la biblioteca (`void BuscarPorTitulo (string t);`)
6) Conocer el número de libros almacenados en la biblioteca (`int Numero ();`)

Cualquier otra operación que sea necesaria para completar el funcionamiento de las clases deberá ser considerada como privada a las mismas.

Además de estas dos clases, se debe construir un programa principal que gestione todo el proceso y que permita que el funcionamiento sea idéntico al de la práctica 1.

La organización del código se debe realizar de manera que las clases tengan asociado un archivo (`.h`) para su interfaz y otro archivo (`.cpp`) para su implementación. Por su parte, el programa principal también estará en un archivo independiente. Por lo que, en total se deberá construir y manejar un proyecto de programación con 5 archivos (`libro.h`, `libro.cpp`, `biblioteca.h`, `biblioteca.cpp`, `pr02.cpp`).

2. COMPILACIÓN SEPARADA EN DEV-C++

Para realizar un programa con diferentes módulos en *Dev-C++* se utiliza el concepto de '*proyecto*'.

En *Dev-C++,* un proyecto es un conjunto de ficheros que una vez compilados conformarán un único programa ejecutable.

La manera de crear un proyecto en el entorno *Dev-C++* es utilizando la opción '*New Project*' que aparece en el menú '*File*'.

Una vez seleccionada la opción nos pedirá qué tipo de proyecto queremos y el nombre del proyecto y el nombre del fichero en el que queremos guardar el proyecto. En este curso nos limitaremos a realizar proyectos de tipo '*consola*' (*Console aplication*).

A partir de aquí, *Dev-C++* abre un proyecto que contiene una sola unidad o módulo con el nombre '*Untitled1*' (o "*main.cpp*" en la versión 4.9.9.2), y que contiene un esbozo de un programa principal.

Podemos utilizar esta primera unidad para empezar a realizar nuestro programa o podemos añadir o eliminar unidades utilizando el botón '*Project*' del menú o pinchando con el botón de la derecha sobre el árbol de proyecto desplegado de la izquierda del programa.

Aunque no es necesario que los ficheros '`.h`' estén incluidos en el proyecto, es conveniente para facilitar su escritura y manipulación.

Una vez tengamos escritas las diferentes unidades de que conste nuestro programa podemos utilizar los botones habituales para '*compilar/compile*', '*ejecutar/execute*' o '*compilar y ejecutar/compile & run*' para probar el programa escrito. O utilizar '*rebuild all*' para recompilar todos los módulos escritos del proyecto.

3. ENTREGA DE PROGRAMAS

Cinco días después de realizada la sesión de prácticas se entregará al profesor la siguiente información:

1) Archivo con el programa (**`pr02##.cpp`**)
2) Archivo con el interfaz de la clase `Libro` (**`libro##.h`**)
3) Archivo con la implementación de la clase `Libro` (**`libro##.cpp`**)
4) Archivo con el interfaz de la clase `Biblioteca` (**`biblioteca##.h`**)
5) Archivo con la implementación de la clase `Biblioteca` (**`biblioteca##.cpp`**)

dónde `##` es el número asignado a la pareja (01, 02,...) en cada grupo de prácticas.

Al comenzar la siguiente sesión de prácticas se entregará al profesor el formulario con la documentación definitiva de la práctica reflejando todo el trabajo realizado realmente en la práctica.

Nota Muy Importante

Antes de poder empezar a realizar cualquiera de las prácticas es necesario presentar las hojas de especificación de programas (formulario de documentación de programas) con las tareas que se van a realizar en la práctica, explicando brevemente como van a solucionarse los problemas que se plantean.

ENTREGA DE PROGRAMAS:
Los días 19, 20, 21 de Marzo y 12 y 13 de Abril de 2005, correspondientes con las fechas de realización de prácticas 14, 15 y 16 de Marzo y 7 y 8 de Abril.

ENTREGA DE FORMULARIOS DEFINITIVOS DE DOCUMENTACIÓN:

Los días 6, 12, 13, 14 y 15 de Abril de 2005, correspondientes con las fechas de realización de prácticas 14, 15 y 16 de Marzo y 7 y 8 de Abril.