Ejercicio para el laboratorio del 24S

Entrada/salida con archivos

Para operar con archivos en C++ se utilizan las clases `ifstream` (para lectura) y `ofstream` (para escritura) de la cabecera

```
#include <fstream>
```

Un archivo se abre al pasar su ruta como argumento al constructor del flujo de archivo o al usar el método `open`.

```
ifstream entrada("entrada.txt");
ofstream salida;  // archivo sin abrir
salida.open("salida.txt");
```

Se puede comprobar que el archivo se ha abierto correctamente con el método `is_open()` y se interactúa con él como con la entrada/salida estándar.

```
int valor;
salida << "1 + 2 = " << 1 + 2 << '\n';
entrada >> valor;
```

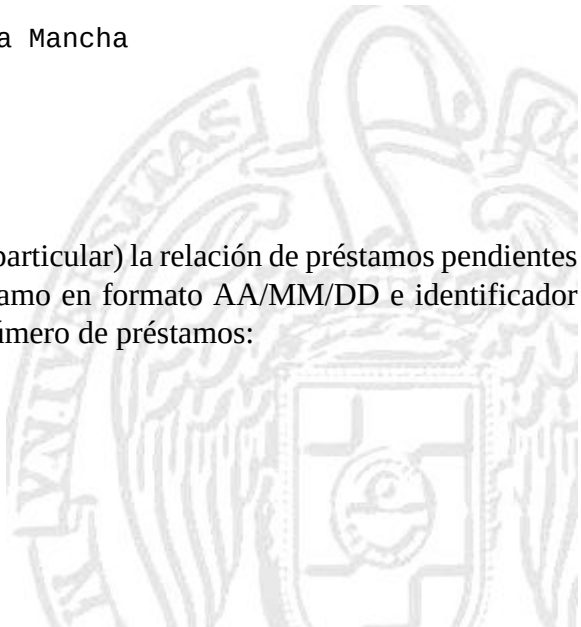
Los archivos se cierran automáticamente cuando su variable sale de ámbito, pero también se pueden cerrar anticipadamente con `.close()`.

Se pide construir un programa que muestre los préstamos pendientes de una biblioteca. Se dispone de dos archivos, uno (`catalogo.txt`) con la información de los ejemplares de los que se dispone la biblioteca. El archivo comienza con un entero indicando el número total de ejemplares, que se describen cada en una línea con un código único, un tipo (libro, audivisual o juego) y un nombre (ordenados de menor a mayor código), separados por espacios como se muestra a continuación:

```
10
1201 L El ingenioso hidalgo don Quijote de la Mancha
1202 A Pulp Fiction
1203 L Yo, robot
1204 J The Legend of Zelda: Echoes of Wisdom
...
```

El otro archivo (`prestamos.txt`) contiene (en ningún orden particular) la relación de préstamos pendientes de devolución (código del ejemplar prestado, fecha de préstamo en formato AA/MM/DD e identificador numérico del usuario). El fichero también comienza con el número de préstamos:

```
7
1204 23/09/24 223
1212 10/09/24 2312
1203 25/08/24 1231
...
```



El programa deberá empezar cargando la información de cada archivo en las estructuras `Catalogo` y `ListaPrestamos`, ambas formadas por un array dinámico de elementos de tipo `Ejemplar` y `Prestamo` respectivamente, su tamaño y un contador indicando el número real de elementos. El catálogo quedará ordenado por orden de códigos (como en el archivo). Una vez leídas las listas, se ordenará la lista de préstamo por fecha de devolución. El programa terminará mostrando la información sobre los préstamo (fecha de devolución, título, días hasta la entrega y penalización por retraso, si la hubiera) como se indica:

```
14/08/24 (en  -41 días) Atraco en alta fidelidad (82 días de penalización)
17/09/24 (en   -7 días) La guerra de los mundos (14 días de penalización)
24/09/24 (en    0 días) Yo, robot
07/10/24 (en   13 días) The Legend of Zelda: Echoes of Wisdom
...
```

Recuerda borrar la memoria dinámica creada.

Tildes en la consola de Windows

En la consola de Windows, para que las tildes, eñes y otros caracteres no ASCII se muestren correctamente es preciso incluir la cabecera `#include <windows.h>` y llamar en el `main` a la función `SetConsoleOutputCP(CP_UTF8)`.[a]

[a] No se debe hacer `using namespace std` antes de `##include <windows.h>` para evitar un conflicto de nombres.

El programa deberá hacer uso de los siguientes subprogramas:

- `leerCatalogo`: carga la información del archivo `catalogo.txt` en el catálogo; devuelve `true` si se ha podido abrir el archivo y `false` en caso contrario. El catálogo sólo contendrá la información de este archivo.
- `buscarEjemplar`: dado el catálogo y un código, devuelve el puntero al ejemplar (de tipo `Ejemplar`) con dicho código; si no se encuentra el código devuelve **`nullptr`**. Debe implementarse como búsqueda binaria.
- `leerPrestamos`: carga la información del archivo `prestamos.txt` en la lista de préstamos; devuelve `true` si se ha podido abrir el archivo y `false` en caso contrario. Cada préstamo debe incluir un campo de tipo `Ejemplar*` con el puntero al ejemplar al que hace referencia el préstamo (que será **`nullptr`** en caso de no encontrarse el código en el catálogo). Para obtener dicho puntero deberás llamar a la función `buscarEjemplar` (ver abajo).
- `ordenarPrestamos`: ordena la lista de préstamo por su fecha de vencimiento (menor a mayor). Como en toda biblioteca, los ejemplares se tienen que devolver algunos días después de haber sido prestados: 30 en el caso de los libros, 7 en el de los materiales audivisuales y 14 en el de los juegos.

  La cabecera `algorithm` de la biblioteca estándar incluye una función `sort` que ordena secuencias de elementos. Recibe tres parámetros: un puntero al comienzo del array, un puntero al primer elemento fuera de él y el nombre de la función que implemente el orden entre elementos. El tercer argumento es opcional y en su defecto se utilizará el operador < que esté definido sobre los elementos (si no hay ninguno dará un error de compilación). Puedes definir el operador < sobre el tipo `Prestamo` con

  ```
  bool operator<(const Prestamo& izdo, const Prestamo& dcho) {
          // Definición del orden
  }
  ```

- `mostrarPrestamos`: dada la lista de préstamos, muestra la relación de préstamos con el formato mostrado arriba.

Como no hemos visto clases ni el uso de múltiples archivos, puedes implementar todo en el mismo fichero fuente .cpp. Eso sí, ten en cuenta que el compilador procesa el fichero de arriba a abajo y por lo tanto no puedes usar tipos ni funciones si no se han definido más arriba en el fichero. En el CV puedes encontrar un ejemplo de ficheros de entrada y un fichero (`salida.txt`) con la salida que tu programa debería generar para dichos ficheros de entrada. También hay incluida una clase `Date` que representa una fecha del calendario y que puedes utilizar en tu programa principal incluyéndola con `#include "Date.hpp"` y añadiendo `Date.cpp` a la lista a archivos a compilar en el proyecto.

Entrega: se debe realizar una entrega por grupo en que solo debéis subir vuestro fichero fuente .cpp (incluyendo vuestros nombres y el identificador del grupo, si ya lo tuviérais). La entrega se realiza en el CV, en la pestaña *Laboratorio y Prácticas*.

Ejercicios adicionales: (1) modifica la estructura (y actualiza todo lo necesario) de manera que la estructura `Catalogo` se represente mediante un array dinámico de punteros a estructuras de tipo `Ejemplar`. (2) Reorganiza el código usando orientación a objetos y separando la declaración de la implementación (aún no lo hemos visto, pero puedes fijarte en la clase `Date`).