VNIVERSITAT DE VALÈNCIA

Laboratorio de PROGRAMACIÓN
Grado en Ingeniería Multimedia (1º)
Curso 2021-22

Práctica Nº 4: Programación Orientada a Objetos

Periodo de realización: Semana del **21/03/2022** al **25/03/2022**

Material a entregar

- **(Repositorio) Proyecto compilable con el código fuente necesario para la práctica.**
- **(Tarea AV) Documentación del código generada por Doxygen.**

Introducción

Una empresa de organización de eventos deportivos nos ha pedido desarrollar una pequeña aplicación para la gestión de diferentes competiciones.

La aplicación deberá permitir a la compañía definir un nuevo evento con la localización (ciudad y país), fecha de celebración, especialidad deportiva, si la competición a organizar es o no internacional y la lista de atletas de alto nivel invitados a la competición.

Una vez definido el evento se deberá poder generar un listado (en fichero) de los atletas a convocar al evento.

Ejercicios

SCV: Debes trabajar en la carpeta "Pr4" del repositorio que ya tienes creado. Descarga los archivos disponibles en Aula Virtual (**Pr4_Fecha.h**, **Pr4_Fecha.cpp** y **Pr4_Prueba.cpp**) y recupera de la práctica 3 los ficheros de la clase Atleta que vas a reutilizar en esta práctica (**Pr3_Atleta.h** y **Pr3_Atleta.cpp**). Haz un un primer *commit* ("Pr.3: Versión inicial") y *push* con este material inicial.

Tarea 1

Se debe comenzar desarrollando la clase `Evento`. Para ello, utiliza como referencia el ejemplo de la clase `Persona` de la práctica 2 o la clase `Atleta` de la práctica 3 y cumple la especificación formal que se indica a continuación:

TAD Evento

Dominio: Representa un evento, caracterizado por la siguiente información:

- código: código del evento.
- nombre: nombre del evento.
- ciudad: ciudad en el que se realizará el evento.

- país: país en el que se realizará el evento.
- fecha: fecha en la que se realizará la competición (un solo día).
- especialidad: especialidad en la que se competirá en el evento (solo una).
- tipo de evento: si el evento es internacional o nacional.
- lista de atletas: listado de atletas invitados al evento.

Operaciones:

- *CrearEvento() -> Evento*
- *AsignarValores(Evento, Cadena, Cadena, Cadena, Cadena, Fecha, Cadena) -> Evento*
- *InvitarAtleta(Evento, Atleta) -> Evento*
- *DevuelveCódigo(Evento) -> Cadena*
- *DevuelveNombre(Evento) -> Cadena*
- *DevuelveLocalización(Evento) -> Cadena, Cadena*
- *DevuelveFecha(Evento) -> Fecha*
- *DevuelveEspecialidad(Evento) -> Cadena*
- *EsInternacional(Evento) -> Booleano*
- *GuardarAtletas(FlujoSalida, Evento)*
- *MostrarEvento(Evento)*

Axiomas:

Sea e ∈ Evento, c1, c2, c3, c4, c5, c6 ∈ Cadena, a ∈ Atleta, fec ∈ Fecha, f ∈ FlujoSalida

CrearEvento()	*Crear un evento. Deberá iniciar todos los campos de información del evento correctamente (las cadenas a "<Sin asignar>", la fecha a un valor correcto de fecha, por ejemplo 1/1/1970, el tipo de evento a internacional y sin atletas invitados al evento).*
AsignarValores(e, c1, c2, c3, c4, fec, c5, c6)	*Asigna valores a la información del evento e:* *c1=código, c2=nombre, c3=ciudad, c4=país, fec=fecha, c5=especialidad,* *si c6='Internacional' el tipo de evento será internacional y si no, nacional*
InvitarAtleta(e, a)	*Añade el atleta* a *a la lista de atletas invitados al evento e.*
DevuelveCódigo(e)	*Devuelve el código del evento e.*
DevuelveNombre(e)	*Devuelve el nombre del evento e.*
DevuelveLocalización(e)	*Devuelve la ciudad y el país donde se celebrará el evento e.*
DevuelveFecha(e)	*Devuelve la fecha de celebración del evento e.*
DevuelveEspecialidad(e)	*Devuelve la especialidad de la competición del evento e.*
EsInternacional(e)	*Devuelve verdadero si el evento es internacional y falso si el evento es nacional.*
GuardarAtletas(f, e)	*Guarda en el **flujo de salida** f los nombres de los atletas invitados al evento e.*
MostrarEvento(e)	*Muestra por pantalla toda la información del evento e.*

La nueva clase se implementará en dos archivos. Un primer archivo, llamado "`Pr4_Evento.h`", con el interfaz de la clase y el segundo, llamado "`Pr4_Evento.cpp`", con la implementación de las operaciones.

Para guardar la lista de atletas invitados al evento utilizaremos *arrays* de C++. Por ello habrá que declarar en la parte privada de la clase Evento los elementos necesarios. Esto es: una constante, un tipo básico de tipo *array* para guardar elementos de tipo Atleta y un registro Atletas que contenga tanto el array con la información de los atletas como un entero con el número de atletas guardados de forma efectiva en el *array*.

```
static const int MAX = 350;
typedef Atleta V_Atletas[MAX];

struct Atletas
{
    V_Atletas info;
    int num;
};
```

Por otro lado, el método *MostrarEvento(e)* se debe implementar con la sobrecarga del operador '**<<**'. Fíjate en la sobrecarga presente en la clase Fecha para hacerlo.

Se debe reutilizar la clase `Atleta` implementada en la práctica 3 (ficheros "`Pr3_Atleta.h`" y "`Pr3_Atleta.cpp`"). No es necesario modificar esta clase para utilizarla en esta práctica.

También deberás utilizar la clase `Fecha` que puedes encontrar en Aula Virtual ("`Pr4_Fecha.h`" y "`Pr4_Fecha.cpp`"). Esta clase no puede modificarse de ninguna manera.

Crea un proyecto en Dev-C++ ("`Pr4_Prueba.dev`") y añade los seis ficheros correspondientes a las clases y el fichero "`Pr4_Prueba.cpp`" y comprueba el correcto funcionamiento de la nueva clase.

Al ejecutar el programa deberías obtener las siguientes pantallas:

```
*** Comprobacion del constructor por defecto ***
***  (y de la sobrecarga del operador '<<')  ***

<Sin asignar> <Sin asignar> (<Sin asignar>)
  <Sin asignar> (<Sin asignar>) - 01/01/1970 - (Internacional)
------------------------------------------------------------
------------------------------------------------------------

Presione una tecla para continuar . . .
```

```
*** Comprobacion del metodo AsignarValores ***

0001 Campeonato universitario de atletismo (5000m (H))
  Valencia (ESP) - 09/03/2022 - (Nacional)
------------------------------------------------------------
------------------------------------------------------------

Presione una tecla para continuar . . .
```

```
*** Comprobacion del metodo InvitarAtleta ***
```

```
0001 - Campeonato universitario de atletismo (5000m (H))
  Valencia (ESP) - 09/3/2022 – (Nacional)
-------------------------------------------------------------
GARONA BURCIU, BIANCA DENISA (ESP)
CEDRAN LLAMUSI, DANIELA BEATRIZ (ESP)
-------------------------------------------------------------

Presione una tecla para continuar . . .
```

```
*** Comprobacion de los 'getters' ***

Codigo: 0001
Nombre: Campeonato universitario de atletismo
Localizacion: Valencia (ESP)
Fecha: 09/3/2022
Especialidad: 5000m (H)
Tipo de evento: Nacional.

Presione una tecla para continuar . . .
```

```
*** Comprobacion del metodo GuardarAtletas ***
Metodo GuardarAtletasEnSalida en pantalla...

Lista de atletas invitados al evento:
-------------------------------------------------------------
O47885;GARONA BURCIU, BIANCA DENISA;ESP;5000m (M)
YI409363;CEDRAN LLAMUSI, DANIELA BEATRIZ;ESP;5000m (M)
-------------------------------------------------------------

Metodo GuardarAtletasEnSalida en fichero...

Atletas invitados guardados correctamente.
```

Con la última comprobación, debería haberse creado también un fichero en tu directorio de trabajo llamado "0001.info" con el siguiente contenido:

Fichero 0001.info

```
O47885;GARONA BURCIU, BIANCA DENISA;ESP;5000m (M)
YI409363;CEDRAN LLAMUSI, DANIELA BEATRIZ;ESP;5000m (M)
```

SCV: Registra los cambios de "Pr4_Prueba.dev", "Pr4_Evento.cpp" y "Pr4_Evento.h" (*commit*) con el mensaje "Pr.4: Completada Tarea 1". **Es MUY RECOMENDABLE realizar** registros intermedios a medida que se van completando grupos de métodos o funciones. En ese caso, utiliza un texto de mensaje que informe sobre el estado de la tarea. P.ej., "Pr.4: Tarea 1. Completado interfaz de la clase Evento" o "Pr.4: Tarea 1. Implementados los métodos 'geters' de la clase Evento".

Tarea 2

Una vez comprobado el correcto funcionamiento de la clase Evento, debemos realizar el programa que nos ha encargado la empresa de organización de eventos deportivos.

Para ello empezaremos creando un nuevo proyecto en Dev-C++ (“`Pr4.dev`”) al que debes añadir los seis ficheros correspondientes a las clases y un nuevo fichero “`Pr4.cpp`”.

Incluye las cabeceras de las tres clases que vamos a utilizar en el nuevo fichero.

La primera tarea que debe realizar el programa es cargar en memoria el listado de deportistas presente en el fichero "atletas_big.csv". Esta tarea ya se hizo en la práctica 3 y solo necesitas recuperar las estructuras de datos y funciones necesarias presentes en esa práctica.

Realiza un programa principal que se limite a llamar a la función "`LeerFichero`" (de la práctica anterior) que lea la información del fichero y muestre a continuación el número de atletas leídos del fichero. El programa principal debe mostrar un mensaje de "Imposible leer el fichero de atletas" si el número de atletas leído es cero.

Compila y ejecuta el programa y comprueba que, efectivamente, se han leído los 300 atletas presentes en el fichero (recuerda que la primera línea del fichero especifica la cabecera de formato y no contiene datos).

SCV: Registra los cambios de “`Pr4.cpp`” (*commit*) con el mensaje “Pr.4: Completada Tarea 2”. Al igual que la tarea anterior, puede ser recomendable realizar registros intermedios a medida que se van completando subtareas o comprobando la correcta ejecución del código escrito.

Tarea 3

A partir de la lista de atletas la empresa debe poder crear un nuevo evento.

Por ello, si el número de atletas leído del fichero es distinto de cero, el programa deberá pedir al usuario los datos necesarios para completar la información del evento.

Crea para esta tarea una nueva función "`IniciarEvento`" que pida la información necesaria y nos devuelva un evento debidamente rellenado.

Se deberá pedir al usuario, dentro de la función, el código y nombre del evento, la ciudad y país donde se celebrará, la fecha, la especialidad y si es o no un evento internacional. Recuerda que el código será siempre una palabra y el país es una palabra conformada por las tres primeras letras (en mayúsculas) del nombre del país. Tanto el nombre del evento, como la ciudad y la especialidad son cadenas de caracteres que pueden contener espacios en blanco.

Con la información pedida al usuario se debe iniciar correctamente un evento nuevo.

Desarrolla las funciones auxiliares que creas conveniente para completar esta tarea.

Verifica el correcto funcionamiento de esta función ejecutando el programa y mostrando adecuadamente, desde el programa principal, el evento por pantalla.

SCV: Registra los cambios de “`Pr4.cpp`” (*commit*) con el mensaje “Pr.4: Completada Tarea 3”. Al

igual que en tareas anteriores, puede ser recomendable realizar registros intermedios a medida que se van completando subtareas.

Tarea 4

Una vez iniciado el evento, el programa cargará en la lista de atletas del evento los atletas que hay que invitar.

Para ello realizaremos una función "`CargarAtletasEvento`" que cargará en la lista de atletas del evento aquellos atletas del listado completo que tengan asignada la especialidad del evento y sean del país en el que se celebra la competición si el evento es nacional o de cualquier país si el evento es internacional.

Desarrolla las funciones auxiliares que creas conveniente para completar esta tarea.

Una vez cargada la lista de invitados muestra todo el evento por pantalla.

Prueba tu programa con las siguientes entradas de datos:

Prueba 1

Codigo: **`Val1.5kH.01`**
Nombre: **`Campeonato Universitario 1500m`**
Ciudad: **`Valencia`**
Pais: **`ESP`**
Fecha (dia, mes y anyo separado por espacios o '/'): **`30/03/2022`**
Especialidad: **`1500m (H)`**
Internacional (S/N)? **`N`**

Con esta información, deberías obtener el siguiente resultado:

```
Val1.5kH.01 Campeonato Universitario 1500m (1500m (H))
  Valencia (ESP) - 30/03/2022 - (Nacional)
------------------------------------------------------------
ORDONO BROCAL, ABDENAJI (ESP)
FIGUEIRAS LLUSIA, BALDOMERO JOSE (ESP)
ESCARPA CHIVA, BRIAN WILLIAM (ESP)
------------------------------------------------------------
```

Prueba 2

Codigo: **`Val1.5kH.02`**
Nombre: **`Campeonato Universitario 1500m`**
Ciudad: **`Valencia`**
Pais: **`ESP`**
Fecha (dia, mes y anyo separado por espacios o '/'): **`31 03 2022`**
Especialidad: **`1500m (M)`**
Internacional (S/N)? **`S`**

Con esta información, deberías obtener el siguiente resultado:

```
Val1.5kH.02 Campeonato Universitario 1500m (1500m (M))
  Valencia (ESP) - 31/03/2022 - (Internacional)
-------------------------------------------------------------
GRECO MAITA, AIME (GBR)
MOMBLONA COMINO, ALAZNE (SUE)
SIENDONES SERNEGUET, ALEXANDRA SOFIA (BEL)
OLARIU OLMEDILLA, ANGELA ESTHER (ESP)
QUINATOA DEVESA, AURELIA MIHAELA (BRA)
ZURRON LAMANA, CARME MARIA (MEX)
GUIOT MONTEALEGRE, CHRISTIANNE (AUS)
TATAY BOLUDA, CLAUDE (ITA)
GARSABALL QUINTAIROS, CRISTINA GEORGETA (FRA)
MOMPARLER OBRADOR, CRISTINA SOLEDAD (BRA)
BAILA MOUSSA, EBA (ITA)
-------------------------------------------------------------
```

SCV: Registra los cambios de "`Pr4.cpp`" (*commit*) con el mensaje "Pr.4: Completada Tarea 4". Puede ser recomendable, como se comentó anteriormente, realizar registros intermedios a medida que se van completando subtareas.

Tarea 5

Ahora la compañía necesita generar un fichero en disco con toda la información del evento.

Si se ha sobrecargado correctamente el operador '`<<`', la salida por fichero debería ser muy sencilla ("`f << e`" siendo '`f`' un fichero de tipo '`ofstream`' correctamente abierto y 'e' un evento).

Desarrolla una función "`GuardarEvento`" que abra un fichero, con nombre el código del evento y extensión '.evento', y guarde en él la información del evento.

La función devolverá si se ha podido o no generar el fichero. Evidentemente NO mostrará ningún mensaje por pantalla.

Comprueba ahora el correcto funcionamiento del programa comprobando que con las entradas de prueba de la Tarea 4 se han generado ficheros con el nombre adecuado y el contenido similar a las salidas por pantalla esperadas en la Tarea 4.

SCV: Registra los cambios de "`Pr4.cpp`" (*commit*) con el mensaje "Pr.4: Completada Tarea 5". Como se comentó anteriormente, puede resultar interesante realizar registros intermedios a medida que se van completando subtareas.

Tarea 6

Para terminar, solo nos queda generar el fichero SOLO con la información de los atletas en formato '.csv' separado por ';', tal y como teníamos la información en el fichero original (línea de cabecera, seguida de una línea con TODA la información de cada uno de los atletas inscritos).

Esta tarea es similar a la desarrollada en la tarea 5, pero cambiando el uso del operador '`<<`' por el uso del método '`GuardarAtletasEnSalida`'.

Desarrolla una última función "GuardarListadoAtletas" a la que le pasaremos el evento y guardará en un fichero de nombre el código del evento y extensión '.csv' la información completa de todos los atletas invitados al evento con el mismo formato que el fichero '.csv' original.

SCV: Registra los últimos cambios de "`Pr4.cpp`" (*commit*) con el mensaje "Pr.4: Versión final, completada Tarea 6". Haz *push*.

Tarea 7: Generación de documentación

El código realizado deberá estar correctamente documentado con Doxygen, de acuerdo con la guía de estilo. Una vez realizada la versión final del programa, se deberá ejecutar Doxygen para generar toda la documentación definitiva. La documentación deberá ser generada exclusivamente en formato en *html* (utilizar como referencia, convenientemente personalizado para esta práctica, el archivo de configuración de Doxygen disponible en el Aula Virtual). Solo la documentación deberá subirse a Aula Virtual (como archivo comprimido). **El programa NO se debe subir al AV, ya que será evaluado directamente desde el repositorio de control de versiones**.

La documentación generada con Doxygen NO debe subirse al repositorio de control de versiones Se debe mantener exclusivamente en el ordenador personal.