VNIVERSITAT DE VALÈNCIA

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

Práctica Nº 3: Programación Orientada a Objetos
Periodo de realización: Semana del **07/03/2022** al **11/03/2022**

Material a entregar

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

Introducción

El objetivo de esta práctica es llegar a desarrollar una aplicación (simple) para gestionar la información de los atletas que participan en una competición de atletismo con diferentes especialidades: carreras de velocidad (100m, 200m, 400m), medio fondo (800m, 1500m) y fondo (5000m y 10000m), tanto en modalidad masculina (M) como femenina (F). Para ello, se implementará (desde cero) una clase C++ que represente a los atletas que participan en la competición, cuya información se almacena en un fichero de datos, y posteriormente se completarán las tareas propias de la aplicación que gestiona la competición.

Ejercicios

SCV: Debes trabajar en la carpeta "Pr3" del repositorio que ya tienes creado. Procede como se indicaba en la tarea 1 de las anteriores prácticas, descargando los archivos disponibles en Aula Virtual y haciendo un primer *commit* ("Pr.3: Versión inicial") y *push* con ese material inicial.

Tarea 1

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

TAD Atleta

Dominio: Representa a un atleta, caracterizado por la siguiente información:

- código: código del atleta.
- nombre: nombre y apellidos del atleta.
- país: país de origen del atleta.
- especialidad: especialidad en la que participa el atleta (solo una).

Operaciones:

- *CrearAtleta () -> Atleta*
- *AsignarValores(Atleta,Cadena,Cadena,Cadena,Cadena) -> Atleta*
- *DevuelveCódigo (Atleta) -> Cadena*
- *DevuelveNombre (Atleta) -> Cadena*
- *DevuelvePais (Atleta) -> Cadena*
- *DevuelveEspecialidad (Atleta) -> Cadena*
- *LeerDeFichero (FlujoEntrada) -> Lógico*
- *Mostrar (Atleta)*

Axiomas:

Sea $a \in Atleta, c1,c2,c3,c4 \in Cadena, f \in FlujoEntrada$

CrearAtleta ()	*Crear un atleta. Deberá inicializar todos los campos de información del atleta con el valor "<Sin asignar>".*
AsignarValores(a,c1,c2,c3,c4)	*Asigna la información del atleta a, c1=código, c2=nombre, c3=país, c4= especialidad.*
DevuelveCódigo (a)	*Devuelve el código del atleta a*
DevuelveNombre (a)	*Devuelve el nombre del atleta a*
DevuelvePais (a)	*Devuelve el país del atleta a*
DevuelveEspecialidad (a)	*Devuelve la especialidad del atleta a*
LeerDeFichero (a, f)	*Lee los datos del atleta a del fichero f. Devuelve verdadero si se ha podido leer los datos y falso en caso contrario.*
Mostrar (a)	*Muestra toda la información del atleta a por pantalla, en una sola línea.*

La clase se debe implementar en dos archivos. Un primer archivo, llamado "`Pr3_Atleta.h`", con el interfaz de la clase y el segundo, llamado "`Pr3_Atleta.cpp`", con la implementación de las operaciones.

Crea un proyecto en Dev-C++ ("`Pr3.dev`") y añade el fichero de la función principal ("`Pr3.cpp`"), que se encuentra en el material de la práctica. Añade también los dos ficheros correspondientes a la clase `Atleta`.

Para comprobar el correcto funcionamiento de la clase anterior, implementa las funciones

```
Atletas LeerFichero(string)

void MostrarAtletas(Atletas)
```

que permiten leer un fichero de atletas ("`atletas_small.csv`") y mostrar su contenido.
Fíjate que el tipo *Atletas* ya está definido en el programa proporcionado como material inicial.

SCV: Registra los cambios de "`Pr3.dev`", "`Pr3.cpp`" (*commit*) con el mensaje "Pr.3: Completada Tarea 1". Puede ser recomendable realizar registros intermedios a medida que se van completando grupos de funciones. En ese caso, utiliza un texto de mensaje que informe sobre el estado de la tarea. P.ej., "Pr.3: Tarea 1. Completadas funciones que utilizan la clase Atleta".

Tarea 2

Una vez comprobado el funcionamiento de la clase Atleta, el programa debe obtener la información de las diferentes especialidades en las que participan los atletas, y los diferentes países de origen de los mismos. Para ello, implementa en el programa las siguientes funciones:

```
Especialidades ObtenerEspecialidades(Atletas)
Paises ObtenerPaises(Atletas)
```

que permiten extraer las diferentes especialidades y países de los atletas participantes, haciendo uso de las estructuras de datos *Especialidades* y *Paises* ya definidas en el programa proporcionado. Ten en cuenta que en los vectores de salida de las funciones no puede haber especialidades o países repetidos.

Implementa también las funciones:

```
void MostrarPaises(Paises)
void MostrarEspecialidades(Especialidades)
```

que muestran, respectivamente, el conjunto de especialidades y países obtenidos previamente de manera numerada (cada especialidad/país se imprime precedida de un número, empezando por 1).

Comprueba que las funciones de extracción e impresión funcionan correctamente mediante las opciones 1 y 2 del programa. Cambia el fichero de datos a "`atletas_big.csv`" y realiza las mismas comprobaciones. En este caso, no muestres la información de los atletas (hay demasiados).

SCV: Registra los cambios de "`Pr3.cpp`" (*commit*) con el mensaje "Pr.3: Completada Tarea 2". Al igual que la tarea anterior, puede ser recomendable realizar registros intermedios a medida que se van completando grupos de métodos.

Tarea 3

A partir de la lista de especialidades obtenida, el programa (ya) solicita al usuario que seleccione una de ellas, con el fin de obtener ahora una lista de los atletas que participan en dicha especialidad. Para ello, implementa la función:

```
Atletas ObtenerAtletasEspecialidad(Atletas,string)
```

La función obtiene, a partir del conjunto completo de atletas, un grupo de atletas cuya especialidad es la indicada por parámetro.

Verifica el correcto funcionamiento de esta función ejecutando la opción 3 del programa.

SCV: Registra el cambio de "`Pr3.cpp`" (*commit*) con el mensaje "Pr.3: Completada Tarea 3".

Tarea 4

A partir de la lista de países obtenida, el programa (ya) solicita al usuario que seleccione una especialidad y un país, con el fin de obtener ahora la lista de los atletas del país que participan en esa especialidad. Para ello, implementa la función:

```
Atletas ObtenerAtletasEspecialidadPais(Atletas,string,string)
```

La función obtiene, a partir del conjunto completo de atletas, un grupo de atletas cuya especialidad y país son los indicados por los parámetros.

Comprueba la función anterior ejecutando la opción 4 del programa.

SCV: Registra el cambio de "`Pr3.cpp`" (*commit*) con el mensaje "Pr.3: Versión final, completada Tarea 4". Haz *push*.

Tarea 5: 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.

Banco de pruebas

Para facilitar la validación de los resultados del programa se proporciona la siguiente tabla que resume el fichero de datos proporcionado, indicando el número de atletas por país y especialidad que aparecen en ese fichero:

Cuenta de id															
Etiquetas de fila	10000m (H)	10000m (M)	100m (H)	100m (M)	1500m (H)	1500m (M)	200m (H)	200m (M)	400m (H)	400m (M)	5000m (H)	5000m (M)	800m (H)	800m (M)	Total general
ALE			3	1	2		1		4	3	5	3	1	6	29
ARG		1		4				1		3	3		1	2	15
AUS	2	2	1	2	1	1	2	1	1	1			2		16
BEL			1			1						3			5
BOL		1	1		1			1	1	1		1	1	1	9
BRA	1	3	2	4	1	2	1	3	3		1	2		1	24
CHI	1		1					1			2				5
DIN		1			1				1		2		1	2	8
ESP	5	2	5	4	3	1	4	1			2	2	3	3	35
FRA	3	2	3	1	1	1	2	1	3	5	3	3	2	2	32
GBR	2	3	4	1		1	1		3	4	3	1	4	3	30
GRE	1		1		1			1				3		2	9
HOL		1					2			1					4
HON		1	1								1	1		1	5
ITA		1	5	3	1	2		3	1	2	6	1	1	2	28
MEX				3	2	1	1	2		2		2	1	2	16
POR			1	2				1		1	1				6
SUE	1		1	2	2	1	1	1				1			10
SUI	1	1	2						1	3			1	1	10
URU					2					1			1		4
Total general	17	19	32	27	18	11	15	17	18	27	29	23	19	28	300