UBA – Facultad de Ciencias Exactas y Naturales – Departamento de Computación

Algoritmos y Estructura de Datos I

Segundo cuatrimestre de 2016

26 de septiembre de 2016

TPI - JJOO v1.0

1. Introducción

En el TPE, hemos especificado el comportamiento de una serie de problemas relacionados con los Juegos Olímpicos. Nuestro próximo paso será obtener una implementación en un lenguaje imperativo, para lo cual la especificación original ha sido *especialmente* adaptada.

El objetivo de este trabajo es programar, usando los conceptos de C++ vistos en clase, tipos y operaciones análogos al de los TP anterior, pero con una especificación que tiene en cuenta el uso de clases y características propias del paradigma imperativo.

Deberán utilizar, además, el tipo abstracto `Lista`, provisto por la cátedra. Publicaremos la interfaz del tipo y su implementación. Para hacer uso de él, sólo deben incluir el archivo .h correspondiente. No hay un .cpp. NO deben mirar la implementación. Sólo deben manejarlo en base a su especificación.

También deberan utilizar fuertemente el tipo `pair` provisto por la `std`. Pueden encontrar una buena referencia para su uso en `http://www.cplusplus.com/reference/std/utility/pair/`

No está permitido utilizar el comando `for`.

En la página de la materia estarán disponibles los archivos mencionados arriba, los *headers* de las clases que deberán implementar y la especificación de cada uno de sus métodos. **Importante: Utilizar la especificación diseñada para este TP, no la solución del TPE!**

Pueden agregar los métodos auxiliares que necesiten. Estos deben estar *en todos los casos* en la parte privada de la clase. No se permite modificar la parte pública de las clases ni agregar atributos adicionales, ya sean públicos o privados. No se permiten archivos "extra" con funciones auxiliares.

2. Demostraciones

Deben implementar la función `atletaProdigio` dentro del archivo ojota.cpp y demostrar su terminación y correctitud. De utilizar funciones auxiliares para su implementación, si las mismas **NO** se encuentran como problemas en la solución de la cátedra, deberán especificarlas y demostrar terminación y correctitud de ellas también.

`problema atletaProdigio` (j: JJOO) = `res` : Atleta {
  `requiere algunaVezSeCompitio` : $|competenciasConOroEnPodio(j)| > 0$;
  `asegura` $esCampeon(result, j)$;
  `asegura` $(\forall c \in competenciasConOroEnPodio(j))anoNacimiento(campeon(c)) \leq anoNacimiento(result)$;
  `aux esCampeon` (a: Atleta, j: JJOO) : Bool = $(\exists c \in competenciasConOroEnPodio(j))a == campeon(c)$;
  `aux campeon` (c: Competencia) : Atleta = $ranking(c)_0$;
  `aux competenciasConOroEnPodio` (j: JJOO) : [Competencia] =
    $[c|c \leftarrow competencias(j), finalizada(c) \wedge |ranking(c)| > 0]$;
}

Observación: De ser necesario, tomar la especificación de las funciones auxiliares del documento provisto por la materia.

3. Implementación de tipos

El tipo `Atleta` mantiene en los atributos `_nombre`, `_sexo`, `_anioNacimiento`, `_nacionalidad` y `_ciaNumber` los observadores del tipo que se llaman de manera similar. En el caso del atributo `_deportes`, mantiene una lista de pares ordenados, cuya primer componente es el deporte y la segunda su capacidad. Dicha lista **DEBE MANTENERSE SIEMPRE ORDENADA** utilizando como criterio el deporte y NO puede tener deportes repetidos. Las capacidades deben estar siempre entre 0 y 100.

El tipo `competencia` posee los atributos `_categoria`, `_participantes` y `_finalizada` que mantienen la información correspondiente a los observadores de mismo nombre. En el atributo `_participantes`, se guardan los participantes. En esta lista no puede haber repetidos. En el atributo `_ranking` se almacenan los ciaNumber de los atletas del observador ranking. Este atributo tampoco puede tener elementos repetidos. Para finalizar, en el atributo `_controlAntidoping` se guarda la información de los observadores lesTocoControlAntiDoping y leDioPositivo utilizando una lista de pares ordenados, donde cada par representa el ciaNumber del atleta al cual le tocó el control y el resultado del mismo. En esta lista no puede haber ciaNumbers repetidos, y todos ellos tienen que pertenecer a algún atleta del atributo `_participantes`.

El tipo `jjoo` almacena sus observadores en los atributos `_anio`, `_atletas`, `_jornadaActual` y `_competenciasPorDia`. En la iésima posición de `_competenciasPorDia`, se encuentra la lista de competencias del día $i+1$. La cantidad de días totales de los juegos se puede obtener a partir de la longitud de la lista del atributo `_competenciasPorDia`.

4. Entrada/Salida

Todas las clases del proyecto tienen tres métodos relacionados con entrada salida:

mostrar : que se encarga de mostrar todo el contenido de la instancia de la clase en el flujo de salida indicado. El formato es a gusto del consumidor, pero esperamos que sea algo más o menos informativo y legible.

guardar : que se encarga de escribir la información de cada instancia en un formato predeterminado que debe ser decodificable por el método que se detalla a continuación (`cargar`).

cargar : que se encarga de leer e interpretar información generada por el método anterior, modificando el valor del parámetro implícito para que coincida con aquel que generó la información.

En definitiva, `guardar` se usará para "grabar" en un archivo de texto el contenido de una instancia mientras que "cargar" se usará para "recuperar" dicha información. En todos los casos, sus interfaces serán:

- `void mostrar(std::ostream& ) const;`
- `void guardar(std::ostream& ) const;`
- `void cargar(std::istream& );`

El detalle del formato que se debe usar para "guardar" y "leer" en cada clase se indica a continuación. Tener en cuenta que los números se deben guardar como texto. Es decir, si escriben a un archivo y después lo miran con un editor de texto, donde escribieron un número 65 deben ver escrito 65 y no una letra A. Cada vez que aparezca un string, éste deberá ir guardado entre `|`. Pueden suponer que los nombres no contendrán el caracter `|`.

Atleta: Se debe guardar el `ENCABEZADO_ARCHIVO` y entre barras (barrita vertical `"|"`) el nombre, el sexo, el año y el país de nacimiento y el ciaNumber. Luego debe almacenarse la lista de los deportes que practica, con sus respectivas capacidades. Por ejemplo, un atleta llamado Liu Song, nacido en China en 1972, que practica sólo el deporte Tenis de Mesa (y su capacidad es 90), debería guardarse de la siguiente manera:
`A |Liu Song| |Masculino| 1972 |China| 123 [(|Tenis de Mesa|, 90)]`
Aclaración 1: en este tipo, y en todos los que involucran listas, no está permitido usar mostrar de Lista para guardar. Sí está permitido usarlo para mostrar.
Las listas deben ir encerradas entre corchetes y cada elemento de la misma debe ir guardado entre paréntesis y separado por comas.
En el caso de las listas donde los elementos sean tuplas, no es necesario agregar paréntesis adicionales.
En el caso de las listas donde los elementos sean números enteros, no es necesario agregar paréntesis.
Aclaración 2: Recordar que en el tipo Atleta es importante respetar el orden de los deportes.

Competencia: Se debe guardar el `ENCABEZADO_ARCHIVO`, la categoría, el estado (si está finalizada o no). Luego deben guardarse la lista de participantes, el ranking (sólo el ciaNumber) y la lista de los atletas a los que les tocó el control antidoping (sólo el ciaNumber) con sus respectivos resultados. A continuación, un ejemplo de Competencia:

```
C (|Rugby|, |Masculino|) |True|
[(A |Juan| |Masculino| 1920 |Argentina| 1 [(|Football|, 35), (|Rugby|, 10)]),
(A |Jorge| |Masculino| 1930 |Argentina| 2 [(|Football|, 32), (|Rugby|, 20)]),
(A |Jackson| |Masculino| 1935 |Escocia| 6 [(|Basket|, 25), (|Football|, 40), (|Rugby|, 5)])]
[1, 6] [(1, |True|), (6, |True|)]
```

En el caso de que la competencia no esté finalizada, las listas de ranking y antidoping deben ser vacías. Por ejemplo:

```
C (|Rugby|, |Masculino|) |False|
[(A |Juan| |Masculino| 1920 |Argentina| 1 [(|Football|, 35), (|Rugby|, 10)]),
(A |Jorge| |Masculino| 1930 |Argentina| 2 [(|Football|, 32), (|Rugby|, 20)]),
(A |Jackson| |Masculino| 1935 |Escocia| 6 [(|Basket|, 25), (|Football|, 40), (|Rugby|, 5)])]
[] []
```

Aclaración 4: Los saltos de línea están puestos solo por razones de legilibilidad en el enunciado. En la implementación debería ser todo una única linea con los saltos de líneas reemplazados por espacios. Por ejemplo:

```
C (|Rugby|, |Masculino|) |False|
[(A |Juan| |Masculino| 20 |Argentina| 1 [] ]
```

```
[] []
```

En realidad se trata de:

```
C (|Rugby|, |Masculino|) |False| [(A |Juan| |Masculino| 20 |Argentina| 1 []] [] []
```

JJOO: Se debe guardar el `ENCABEZADO_ARCHIVO`, el año, la jornada actual, la lista de participantes, y la lista con las listas de competencias de cada día, ordenadas por día (en la cabeza de la lista, la lista de competencias del primer día y así siguiendo).

```
J 2016 3
[(A |Juan| |Masculino| 1920 |Argentina| 1 [(|Football|, 35), (|Rugby|, 10)]),
(A |Jorge| |Masculino| 1930 |Argentina| 2 [(|Football|, 32), (|Rugby|, 20)]),
(A |Juliana| |Femenino| 1932 |Bosnia| 3 [(|Handball|, 50)]),
(A |Jenny| |Femenino| 1922 |USA| 4 [(|Softball|, 20)]),
(A |Jordan| |Masculino| 1930 |USA| 5 [(|Basket|, 25), (|Football|, 40), (|Rugby|, 5)]),
(A |Jackson| |Masculino| 1935 |Escocia| 6 [(|Basket|, 25), (|Football|, 40), (|Rugby|, 5)])]
[[(C (|Rugby|, |Masculino|) |True|
[(A |Juan| |Masculino| 1920 |Argentina| 1 [(|Football|, 35), (|Rugby|, 10)]),
(A |Jorge| |Masculino| 1930 |Argentina| 2 [(|Football|, 32), (|Rugby|, 20)]),
(A |Jackson| |Masculino| 1935 |Escocia| 6 [(|Basket|, 25), (|Football|, 40), (|Rugby|, 5)])]
[2, 1, 6] [(2, |True|)]),
(C (|Football|, |Masculino|) |True|
[(A |Juan| |Masculino| 1920 |Argentina| 1 [(|Football|, 35), (|Rugby|, 10)]),
(A |Jorge| |Masculino| 1930 |Argentina| 2 [(|Football|, 32), (|Rugby|, 20)])]
[1, 2] [(1, |True|)])],
[],
[(C (|Handball|, |Femenino|) |False|
[(A |Juliana| |Femenino| 1932 |Bosnia| 3 [(|Handball|, 50)])] [] []),
(C (|Softball|, |Femenino|) |False| [] [] [])],
[]]
```