Formato de entrega

El formato de entrega del obligatorio va a constar de un archivo zip que en su interior contenga únicamente archivos con extensiones “.cpp” y “.h” en caso de C/C++ o “.java” en caso de utilizar Java. También podrán existir directorios para organizar mejor el código y poder reutilizar implementaciones en varios ejercicios.

C/C++

En el caso de C/C++, la estructura dentro del zip tendrá el siguiente formato:

```
├── directorioA
├── directorioB
...
├── directorioZ
├── ejercicio1.cpp
├── ejercicio2.cpp
...
└── ejercicioX.cpp
```

Por ejemplo, si tuviéramos 4 ejercicios, un posible zip a entregar podría tener:

```
├── funciones
│   └── enteros.cpp
├── tads
│   ├── avl.cpp
│   └── avl.h
├── ejercicio1.cpp
├── ejercicio2.cpp
├── ejercicio3.cpp
└── ejercicio4.cpp
```

Notar que es obligatorio que en la raíz del zip se encuentren los archivos “ejercicioX.cpp”. Los directorios para funciones auxiliares y TADs pueden llamarse de cualquier forma y no son obligatorios. Un ejemplo donde no se utilizan directorios auxiliares podría ser:

```
├── enteros.cpp
├── avl.cpp
├── avl.h
├── ejercicio1.cpp
├── ejercicio2.cpp
├── ejercicio3.cpp
└── ejercicio4.cpp
```

Cómo compilar

La forma de compilar cada ejercicio es la vista en el curso, pasando por parámetro el archivo raíz “ejercicioX.cpp”, siendo X el número de ejercicio, e indicando en el parámetro “-o” el nombre del archivo compilado:
`g++ -o ejercicioX ejercicioX.cpp`

Cómo realizar las inclusiones

Se puede ver un ejemplo del formato y de cómo se realizan las inclusiones en este link.

Las inclusiones parten del archivo “raíz” del programa, es decir, el “ejercicioX.cpp”. Dependiendo de si los archivos a incluir se encuentran en el mismo directorio o no, es cómo se realizará la inclusión.

Con directorios

Dado el siguiente caso:

```
├── funciones
│   └── enteros.cpp
├── tads
│   ├── avl.cpp
│   └── avl.h
..
```

Se definen las inclusiones de la siguiente manera:

```
#include "tads/avl.h"
#include "tads/avl.cpp"
#include “funciones/enteros.cpp”
```

Sin directorios

Dado el caso en el que no se usan directorios:

```
├── enteros.cpp
├── avl.cpp
├── avl.h
...
```

Se definen las inclusiones de la siguiente manera:

```
#include "avl.h"
#include "avl.cpp"
#include "enteros.cpp"
```

Aclaración sobre TADs

Notar que en el archivo raíz se incluye tanto la especificación del TAD (avl.h) como la implementación (avl.cpp). Lo correcto es incluir únicamente la especificación y asignar la implementación al momento de compilar. Esto resulta en problemas para

realizar debugging y en que la compilación sea más compleja de estandarizar, por lo que se decide que es aceptable realizar la inclusión de la implementación.

JAVA

En el caso de Java, la estructura dentro del zip tendrá el siguiente formato:

```
├── directorioA
├── directorioB
...
├── directorioZ
├── Ejercicio1.java
├── Ejercicio2.java
...
└── EjercicioX.java
```

Por ejemplo, si tuviéramos 4 ejercicios, un posible zip a entregar podría tener:

```
├── funciones
│   └── Enteros.java
├── tads
│   └── AVL.java
├── Ejercicio1.java
├── Ejercicio2.java
├── Ejercicio3.java
└── Ejercicio4.java
```

Notar que es obligatorio que en la raíz del zip se encuentren los archivos “EjercicioX.java”. Los directorios para funciones auxiliares y TADs pueden llamarse de cualquier forma y no son obligatorios. Un ejemplo donde no se utilizan directorios auxiliares podría ser:

```
├── Enteros.java
├── AVL.java
├── Ejercicio1.java
├── Ejercicio2.java
├── Ejercicio3.java
└── Ejercicio4.java
```

Cómo compilar

La forma de compilar cada ejercicio es la vista en el curso, pasando por parámetro el archivo raíz “EjercicioX.java”, siendo X el número de ejercicio:
`javac EjercicioX.java`

Cómo realizar las importaciones

Se puede ver un ejemplo del formato y de cómo se realizan las importaciones en este link.

Las importaciones parten del archivo “raíz” del programa, es decir, el “EjercicioX.java”.
Dependiendo de si los archivos a incluir se encuentran en el mismo directorio o no, es cómo se realizan.

Con directorios

Dado el siguiente caso:

```
├── funciones
│   └── Enteros.java
├── tads
│   └── AVL.java
..
```

Se definen las importaciones de la siguiente manera:

```
import tads.AVL;
import funciones.Enteros;
```

Para que esto funcione es importante definir los paquetes java de las clases que se encuentran en esos directorios.
Dado el ejemplo anterior, la clase Enteros.java debe definir su paquete al comienzo del archivo:

```
package funciones;
```

De la misma forma lo debe hacer la clase AVL.java:

```
package tads;
```

Las clases de la raíz, “EjercicioX.java” por ejemplo, no deben tener definido un paquete.

Sin directorios

Dado el caso en el que no se usan directorios:

```
├── Enteros.java
├── AVL.java
..
```

No es necesario definir ninguna importación. Es importante que ninguna clase del directorio raíz tenga definido un paquete al comienzo de la misma.