SIMULACIÓN DE ACCESO A UN EVENTO DEPORTIVO (2)

Alberto Larena Luengo

Alba Calvo Herrero

PECL PARTE 2 Estructuras de Datos

Contenido

1. Introducción 2
2. Capturas de pantalla de un ejemplo realizado paso a paso. 2
2. Detalles y justificación de la implementación: 13
2.1. Especificación concreta de la interfaz de los TAD´s implementados 13
2.1.1. TAD`s creados y definición operaciones 13
2.2. Solución adoptada: descripción de las dificultades encontradas. 20
2.3. Diseño de la relación entre las clases de los TAD implementados. 21
2.3.1. Diagrama UML. 21
2.3.2 Explicación de los métodos más destacados. 21
2.4. Explicación del comportamiento del programa. 27
2.5 Bibliografía 29

1. Introducción

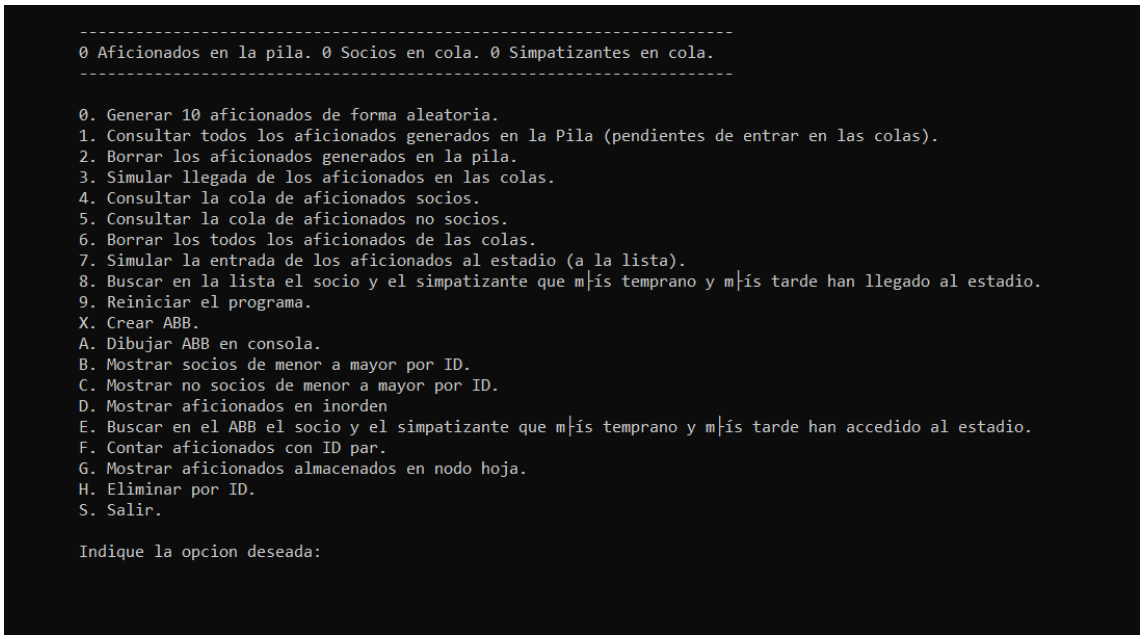
Este documento es la parte de documentación del programa de simulación de acceso a un evento deportivo.

Como esta segunda parte del trabajo es la ampliación del mismo programa hemos decidido hacer lo mismo con la documentación. Por lo que encontrará una parte con la PECL de la parte 1 ya realizada anteriormente y otra de la parte 2 en cada apartado. Esta segunda parte vendrá indicada en cada uno de los apartados en los que es necesario para facilitar la comprensión y corrección de la práctica.

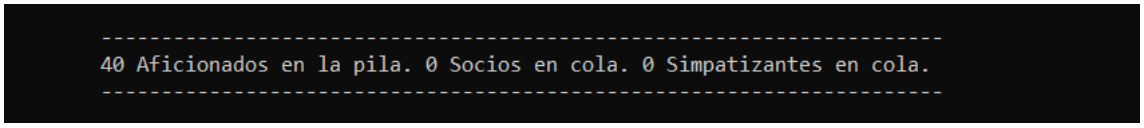
2. Capturas de pantalla de un ejemplo realizado paso a paso.

A continuación se mostrará el funcionamiento del programa mediante un ejemplo con capturas de pantalla.

Al ejecutar el programa aparecerá el menú con las opciones disponibles y en la cabecera se muestra el número de aficionados, los socios y los simpatizantes que se actualiza cada vez que elegimos una opción.



Primero generamos los aficionados pulsando **0**. Los aficionados tienen una hora de llegada (entre las 18:00 y las 18:59) y un identificador único. Ambos son generados aleatoriamente. En el caso del identificador, la primera vez que use esta opción será un número entre el 1 y el 10, la segunda entre el 11 y el 20, y así sucesivamente. Además, serán socios o simpatizantes según su identificador. (Si es par será socio y sino simpatizante). En este ejemplo daremos 0 cuatro veces para generar 40 aficionados que se almacenarán en la pila. Vemos como se actualiza la cabecera.



Si pulsamos **1**, mostrará todos los aficionados que hay en la pila.

```
El aficionado 34    es socio y llego a las 18:43 horas.
El aficionado 36    es socio y llego a las 18: 4 horas.
El aficionado 40    es socio y llego a las 18:23 horas.
El aficionado 39 no es socio y llego a las 18:47 horas.
El aficionado 35 no es socio y llego a las 18:23 horas.
El aficionado 32    es socio y llego a las 18:54 horas.
El aficionado 38    es socio y llego a las 18:31 horas.
El aficionado 37 no es socio y llego a las 18:36 horas.
El aficionado 33 no es socio y llego a las 18:26 horas.
El aficionado 31 no es socio y llego a las 18:48 horas.
El aficionado 29 no es socio y llego a las 18:28 horas.
El aficionado 25 no es socio y llego a las 18: 2 horas.
El aficionado 22    es socio y llego a las 18:28 horas.
El aficionado 23 no es socio y llego a las 18:23 horas.
El aficionado 21 no es socio y llego a las 18: 1 horas.
El aficionado 30    es socio y llego a las 18:45 horas.
El aficionado 24    es socio y llego a las 18:55 horas.
El aficionado 27 no es socio y llego a las 18:20 horas.
El aficionado 26    es socio y llego a las 18:31 horas.
El aficionado 28    es socio y llego a las 18:40 horas.
El aficionado 17 no es socio y llego a las 18: 0 horas.
El aficionado 11 no es socio y llego a las 18: 4 horas.
El aficionado 16    es socio y llego a las 18:16 horas.
El aficionado 12    es socio y llego a las 18:31 horas.
El aficionado 19 no es socio y llego a las 18:33 horas.
El aficionado 18    es socio y llego a las 18:26 horas.
El aficionado 13 no es socio y llego a las 18:15 horas.
El aficionado 14    es socio y llego a las 18:51 horas.
El aficionado 20    es socio y llego a las 18:35 horas.
El aficionado 15 no es socio y llego a las 18:31 horas.
El aficionado  1 no es socio y llego a las 18:53 horas.
El aficionado  3 no es socio y llego a las 18: 0 horas.
El aficionado  9 no es socio y llego a las 18:36 horas.
El aficionado  8    es socio y llego a las 18:13 horas.
El aficionado  7 no es socio y llego a las 18:57 horas.
El aficionado 10    es socio y llego a las 18:30 horas.
El aficionado  2    es socio y llego a las 18:56 horas.
El aficionado  4    es socio y llego a las 18:41 horas.
El aficionado  6    es socio y llego a las 18:42 horas.
El aficionado  5 no es socio y llego a las 18:14 horas.
```

Pulsando **2**, borrará todos los aficionados almacenados en la pila.

```
--------------------------------------------------------------------------
0 Aficionados en la pila. 0 Socios en cola. 0 Simpatizantes en cola.
--------------------------------------------------------------------------
```

Ahora volvemos a generar 40 aficionados sin reiniciar el programa. Esta vez los ID tienen que estar entre 41 y 80.

```
    El aficionado 71 no es socio y llego a las 18:39 horas.
    El aficionado 80    es socio y llego a las 18:23 horas.
    El aficionado 78    es socio y llego a las 18:33 horas.
    El aficionado 73 no es socio y llego a las 18:51 horas.
    El aficionado 76    es socio y llego a las 18:27 horas.
    El aficionado 77 no es socio y llego a las 18:35 horas.
    El aficionado 72    es socio y llego a las 18: 6 horas.
    El aficionado 75 no es socio y llego a las 18:54 horas.
    El aficionado 79 no es socio y llego a las 18:16 horas.
    El aficionado 74    es socio y llego a las 18:54 horas.
    El aficionado 64    es socio y llego a las 18:32 horas.
    El aficionado 61 no es socio y llego a las 18:18 horas.
    El aficionado 69 no es socio y llego a las 18:48 horas.
    El aficionado 63 no es socio y llego a las 18:46 horas.
    El aficionado 65 no es socio y llego a las 18:11 horas.
    El aficionado 70    es socio y llego a las 18:40 horas.
    El aficionado 62    es socio y llego a las 18:16 horas.
    El aficionado 66    es socio y llego a las 18: 1 horas.
    El aficionado 68    es socio y llego a las 18:18 horas.
    El aficionado 67 no es socio y llego a las 18:45 horas.
    El aficionado 51 no es socio y llego a las 18:13 horas.
    El aficionado 58    es socio y llego a las 18:53 horas.
    El aficionado 52    es socio y llego a las 18: 4 horas.
    El aficionado 57 no es socio y llego a las 18:45 horas.
    El aficionado 56    es socio y llego a las 18:43 horas.
    El aficionado 60    es socio y llego a las 18:33 horas.
    El aficionado 59 no es socio y llego a las 18: 4 horas.
    El aficionado 55 no es socio y llego a las 18:44 horas.
    El aficionado 53 no es socio y llego a las 18:49 horas.
    El aficionado 54    es socio y llego a las 18:37 horas.
    El aficionado 44    es socio y llego a las 18:25 horas.
    El aficionado 41 no es socio y llego a las 18:11 horas.
    El aficionado 43 no es socio y llego a las 18:43 horas.
    El aficionado 49 no es socio y llego a las 18:29 horas.
    El aficionado 46    es socio y llego a las 18:31 horas.
    El aficionado 47 no es socio y llego a las 18: 5 horas.
    El aficionado 42    es socio y llego a las 18:16 horas.
    El aficionado 45 no es socio y llego a las 18:15 horas.
    El aficionado 48    es socio y llego a las 18:27 horas.
    El aficionado 50    es socio y llego a las 18:28 horas.
```

Al pulsar la opción **3**, se almacenarán los aficionados en dos colas. Dependiendo de si son socios o no se insertarán en la cola de los socios o de los simpatizantes.

```
 ---------------------------------------------------------------------
0 Aficionados en la pila. 20 Socios en cola. 20 Simpatizantes en cola.
 ---------------------------------------------------------------------
```

Con el **4** mostrará la cola de los aficionados socios

```
Cola de Socios:
        El aficionado 80    es socio y llego a las 18:23 horas.
        El aficionado 78    es socio y llego a las 18:33 horas.
        El aficionado 76    es socio y llego a las 18:27 horas.
        El aficionado 72    es socio y llego a las 18: 6 horas.
        El aficionado 74    es socio y llego a las 18:54 horas.
        El aficionado 64    es socio y llego a las 18:32 horas.
        El aficionado 70    es socio y llego a las 18:40 horas.
        El aficionado 62    es socio y llego a las 18:16 horas.
        El aficionado 66    es socio y llego a las 18: 1 horas.
        El aficionado 68    es socio y llego a las 18:18 horas.
        El aficionado 58    es socio y llego a las 18:53 horas.
        El aficionado 52    es socio y llego a las 18: 4 horas.
        El aficionado 56    es socio y llego a las 18:43 horas.
        El aficionado 60    es socio y llego a las 18:33 horas.
        El aficionado 54    es socio y llego a las 18:37 horas.
        El aficionado 44    es socio y llego a las 18:25 horas.
        El aficionado 46    es socio y llego a las 18:31 horas.
        El aficionado 42    es socio y llego a las 18:16 horas.
        El aficionado 48    es socio y llego a las 18:27 horas.
        El aficionado 50    es socio y llego a las 18:28 horas.
```

y con el **5** la cola de los simpatizantes.

```
Cola de Simpatizantes:
        El aficionado 71 no es socio y llego a las 18:39 horas.
        El aficionado 73 no es socio y llego a las 18:51 horas.
        El aficionado 77 no es socio y llego a las 18:35 horas.
        El aficionado 75 no es socio y llego a las 18:54 horas.
        El aficionado 79 no es socio y llego a las 18:16 horas.
        El aficionado 61 no es socio y llego a las 18:18 horas.
        El aficionado 69 no es socio y llego a las 18:48 horas.
        El aficionado 63 no es socio y llego a las 18:46 horas.
        El aficionado 65 no es socio y llego a las 18:11 horas.
        El aficionado 67 no es socio y llego a las 18:45 horas.
        El aficionado 51 no es socio y llego a las 18:13 horas.
        El aficionado 57 no es socio y llego a las 18:45 horas.
        El aficionado 59 no es socio y llego a las 18: 4 horas.
        El aficionado 55 no es socio y llego a las 18:44 horas.
        El aficionado 53 no es socio y llego a las 18:49 horas.
        El aficionado 41 no es socio y llego a las 18:11 horas.
        El aficionado 43 no es socio y llego a las 18:43 horas.
        El aficionado 49 no es socio y llego a las 18:29 horas.
        El aficionado 47 no es socio y llego a las 18: 5 horas.
        El aficionado 45 no es socio y llego a las 18:15 horas.
```

Si se pulsa la opción **6** se borrarán todos los aficionados almacenados en ambas colas.

La opción **7** extrae los aficionados de ambas colas y los almacena en una lista de forma ordenada. Primero, se extraerán los aficionados de la cola de los socios y se ordenarán en función de la hora de llegada de cada socio. Es decir, primero los que han llegado más pronto. Una vez hecho esto, se extraen los aficionados simpatizantes y también se ordenan según su hora de llegada.

```
        El aficionado 66    es socio y llego a las 18: 1 horas.
        El aficionado 52    es socio y llego a las 18: 4 horas.
        El aficionado 72    es socio y llego a las 18: 6 horas.
        El aficionado 42    es socio y llego a las 18:16 horas.
        El aficionado 62    es socio y llego a las 18:16 horas.
        El aficionado 68    es socio y llego a las 18:18 horas.
        El aficionado 80    es socio y llego a las 18:23 horas.
        El aficionado 44    es socio y llego a las 18:25 horas.
        El aficionado 48    es socio y llego a las 18:27 horas.
        El aficionado 76    es socio y llego a las 18:27 horas.
        El aficionado 50    es socio y llego a las 18:28 horas.
        El aficionado 46    es socio y llego a las 18:31 horas.
        El aficionado 64    es socio y llego a las 18:32 horas.
        El aficionado 60    es socio y llego a las 18:33 horas.
        El aficionado 78    es socio y llego a las 18:33 horas.
        El aficionado 54    es socio y llego a las 18:37 horas.
        El aficionado 70    es socio y llego a las 18:40 horas.
        El aficionado 56    es socio y llego a las 18:43 horas.
        El aficionado 58    es socio y llego a las 18:53 horas.
        El aficionado 74    es socio y llego a las 18:54 horas.
        El aficionado 59 no es socio y llego a las 18: 4 horas.
        El aficionado 47 no es socio y llego a las 18: 5 horas.
        El aficionado 65 no es socio y llego a las 18:11 horas.
        El aficionado 41 no es socio y llego a las 18:11 horas.
        El aficionado 51 no es socio y llego a las 18:13 horas.
        El aficionado 45 no es socio y llego a las 18:15 horas.
        El aficionado 79 no es socio y llego a las 18:16 horas.
        El aficionado 61 no es socio y llego a las 18:18 horas.
        El aficionado 49 no es socio y llego a las 18:29 horas.
        El aficionado 77 no es socio y llego a las 18:35 horas.
        El aficionado 71 no es socio y llego a las 18:39 horas.
        El aficionado 43 no es socio y llego a las 18:43 horas.
        El aficionado 55 no es socio y llego a las 18:44 horas.
        El aficionado 67 no es socio y llego a las 18:45 horas.
        El aficionado 57 no es socio y llego a las 18:45 horas.
        El aficionado 63 no es socio y llego a las 18:46 horas.
        El aficionado 69 no es socio y llego a las 18:48 horas.
        El aficionado 53 no es socio y llego a las 18:49 horas.
        El aficionado 73 no es socio y llego a las 18:51 horas.
        El aficionado 75 no es socio y llego a las 18:54 horas.


        ------------------------------------------------------------------
        0 Aficionados en la pila. 0 Socios en cola. 0 Simpatizantes en cola.
        ------------------------------------------------------------------
```

Pulsando **8** buscará en la lista cuatro aficionados: el primer aficionado en acceder al estadio, el último socio en acceder al estadio y el último aficionado en acceder al estadio.

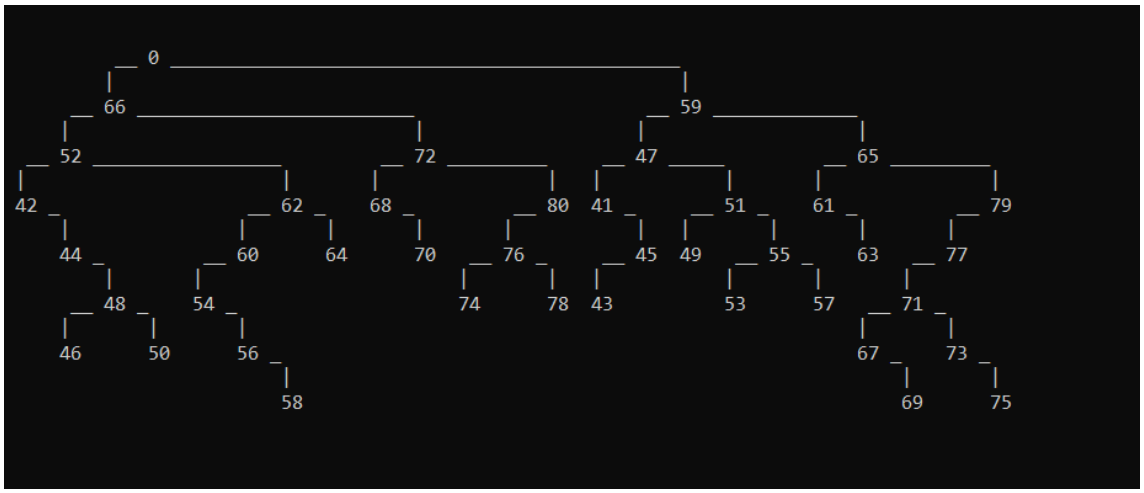
```
El primer socio es:            El aficionado 66    es socio y llego a las 18: 1 horas.
El ├╢ltimo socio es:           El aficionado 74    es socio y llego a las 18:54 horas.
El primer simpatizante es:     El aficionado 59 no es socio y llego a las 18: 4 horas.
El ├╢ltimo simpatizante es:    El aficionado 75 no es socio y llego a las 18:54 horas.
```

Si pulsamos **9** se reiniciará el programa al estado inicial.

A continuación, se verá el funcionamiento de las opciones de la segunda parte de la práctica. La del árbol binario de búsqueda.

Con la opción **X** se creará el ABB (árbol binario de búsqueda) a partir de los aficionados almacenados en la lista creada con la opción 7 vista anteriormente. El nodo raíz del árbol almacenará un aficionado ficticio que es socio (el 0). Los aficionados que son socios se insertarán ordenados por sus IDs en el subárbol izquierdo del nodo raíz y los que son simpatizantes en el subárbol derecho, también ordenados por sus IDs.

Pulsamos **A** para visualizar el árbol.



Con la opción **B** se mostrarán todos los socios ordenados por sus IDs de menor a mayor (sin incluir el aficionado almacenado en nodo ficticio).

```
        El aficionado 42    es socio y llego a las 18:16 horas.
        El aficionado 44    es socio y llego a las 18:25 horas.
        El aficionado 46    es socio y llego a las 18:31 horas.
        El aficionado 48    es socio y llego a las 18:27 horas.
        El aficionado 50    es socio y llego a las 18:28 horas.
        El aficionado 52    es socio y llego a las 18: 4 horas.
        El aficionado 54    es socio y llego a las 18:37 horas.
        El aficionado 56    es socio y llego a las 18:43 horas.
        El aficionado 58    es socio y llego a las 18:53 horas.
        El aficionado 60    es socio y llego a las 18:33 horas.
        El aficionado 62    es socio y llego a las 18:16 horas.
        El aficionado 64    es socio y llego a las 18:32 horas.
        El aficionado 66    es socio y llego a las 18: 1 horas.
        El aficionado 68    es socio y llego a las 18:18 horas.
        El aficionado 70    es socio y llego a las 18:40 horas.
        El aficionado 72    es socio y llego a las 18: 6 horas.
        El aficionado 74    es socio y llego a las 18:54 horas.
        El aficionado 76    es socio y llego a las 18:27 horas.
        El aficionado 78    es socio y llego a las 18:33 horas.
        El aficionado 80    es socio y llego a las 18:23 horas.
```

Con la opción **C** se muestran todos los simpatizantes ordenados por sus IDs de menor a mayor.

```
El aficionado 41 no es socio y llego a las 18:11 horas.
El aficionado 43 no es socio y llego a las 18:43 horas.
El aficionado 45 no es socio y llego a las 18:15 horas.
El aficionado 47 no es socio y llego a las 18: 5 horas.
El aficionado 49 no es socio y llego a las 18:29 horas.
El aficionado 51 no es socio y llego a las 18:13 horas.
El aficionado 53 no es socio y llego a las 18:49 horas.
El aficionado 55 no es socio y llego a las 18:44 horas.
El aficionado 57 no es socio y llego a las 18:45 horas.
El aficionado 59 no es socio y llego a las 18: 4 horas.
El aficionado 61 no es socio y llego a las 18:18 horas.
El aficionado 63 no es socio y llego a las 18:46 horas.
El aficionado 65 no es socio y llego a las 18:11 horas.
El aficionado 67 no es socio y llego a las 18:45 horas.
El aficionado 69 no es socio y llego a las 18:48 horas.
El aficionado 71 no es socio y llego a las 18:39 horas.
El aficionado 73 no es socio y llego a las 18:51 horas.
El aficionado 75 no es socio y llego a las 18:54 horas.
El aficionado 77 no es socio y llego a las 18:35 horas.
El aficionado 79 no es socio y llego a las 18:16 horas.
```

Con la opción **D** los datos de todos los aficionados recorriendo el ABB en <u>inorden</u>.

```
        El aficionado 42    es socio y llego a las 18:16 horas.
        El aficionado 44    es socio y llego a las 18:25 horas.
        El aficionado 46    es socio y llego a las 18:31 horas.
        El aficionado 48    es socio y llego a las 18:27 horas.
        El aficionado 50    es socio y llego a las 18:28 horas.
        El aficionado 52    es socio y llego a las 18: 4 horas.
        El aficionado 54    es socio y llego a las 18:37 horas.
        El aficionado 56    es socio y llego a las 18:43 horas.
        El aficionado 58    es socio y llego a las 18:53 horas.
        El aficionado 60    es socio y llego a las 18:33 horas.
        El aficionado 62    es socio y llego a las 18:16 horas.
        El aficionado 64    es socio y llego a las 18:32 horas.
        El aficionado 66    es socio y llego a las 18: 1 horas.
        El aficionado 68    es socio y llego a las 18:18 horas.
        El aficionado 70    es socio y llego a las 18:40 horas.
        El aficionado 72    es socio y llego a las 18: 6 horas.
        El aficionado 74    es socio y llego a las 18:54 horas.
        El aficionado 76    es socio y llego a las 18:27 horas.
        El aficionado 78    es socio y llego a las 18:33 horas.
        El aficionado 80    es socio y llego a las 18:23 horas.
        El aficionado 41 no es socio y llego a las 18:11 horas.
        El aficionado 43 no es socio y llego a las 18:43 horas.
        El aficionado 45 no es socio y llego a las 18:15 horas.
        El aficionado 47 no es socio y llego a las 18: 5 horas.
        El aficionado 49 no es socio y llego a las 18:29 horas.
        El aficionado 51 no es socio y llego a las 18:13 horas.
        El aficionado 53 no es socio y llego a las 18:49 horas.
        El aficionado 55 no es socio y llego a las 18:44 horas.
        El aficionado 57 no es socio y llego a las 18:45 horas.
        El aficionado 59 no es socio y llego a las 18: 4 horas.
        El aficionado 61 no es socio y llego a las 18:18 horas.
        El aficionado 63 no es socio y llego a las 18:46 horas.
        El aficionado 65 no es socio y llego a las 18:11 horas.
        El aficionado 67 no es socio y llego a las 18:45 horas.
        El aficionado 69 no es socio y llego a las 18:48 horas.
        El aficionado 71 no es socio y llego a las 18:39 horas.
        El aficionado 73 no es socio y llego a las 18:51 horas.
        El aficionado 75 no es socio y llego a las 18:54 horas.
        El aficionado 77 no es socio y llego a las 18:35 horas.
        El aficionado 79 no es socio y llego a las 18:16 horas.
```

Pulsando la opción **E** busca en el ABB y mostrar los siguientes 4 aficionados:

- El primer aficionado en acceder al estadio.
- El último socio en acceder al estadio.
- El primer simpatizante en acceder al estadio.
- El último aficionado en acceder al estadio.

```
Primer aficionado en acceder al estadio:        El aficionado 66    es socio y llego a las 18: 1 horas.
Ultimo socio en acceder al estadio:             El aficionado 74    es socio y llego a las 18:54 horas.
Primer simpatizante en acceder al estadio:      El aficionado 59 no es socio y llego a las 18: 4 horas.
Ultimo aficionado en acceder al estadio:        El aficionado 74    es socio y llego a las 18:54 horas.
```

Si pulsamos **F** contará el número de aficionados almacenados en el ABB en los que los ID’s son pares.

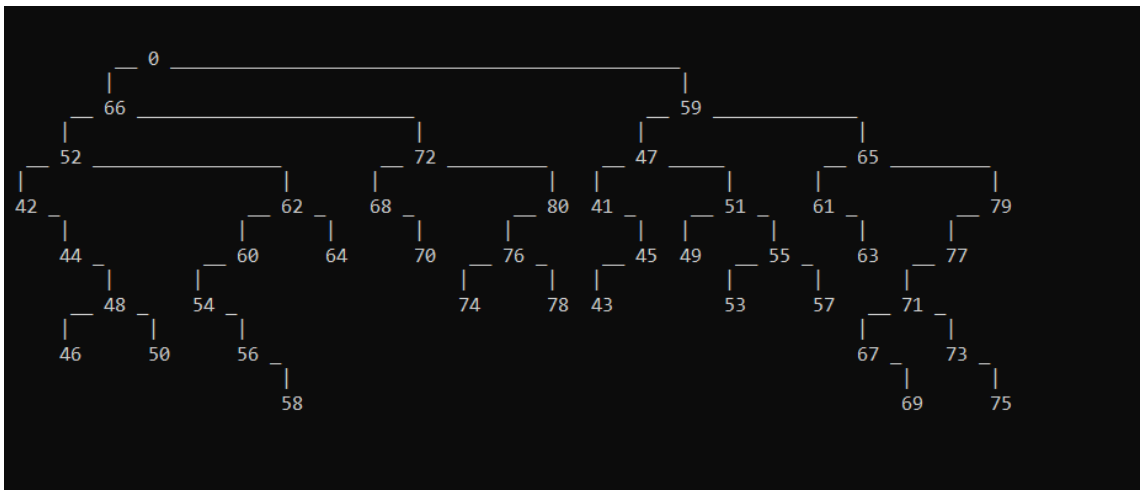
```
Socios en el ABB: 20
```

Con la **G** mostrará los aficionados que se encuentran almacenados en un nodo hoja.

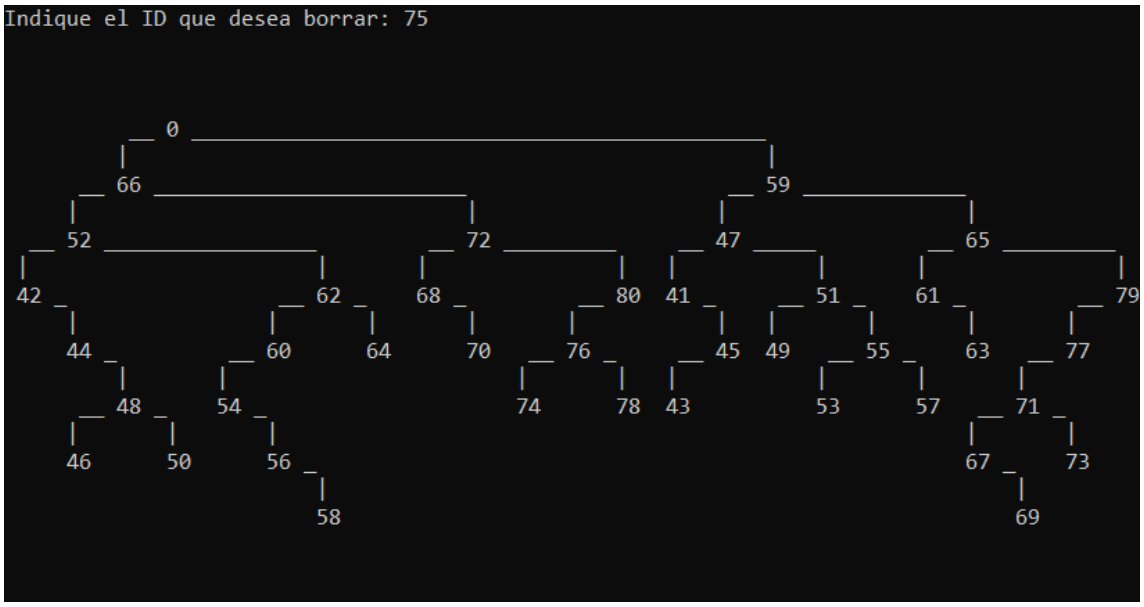
```
        El aficionado 46    es socio y llego a las 18:31 horas.
        El aficionado 50    es socio y llego a las 18:28 horas.
        El aficionado 58    es socio y llego a las 18:53 horas.
        El aficionado 64    es socio y llego a las 18:32 horas.
        El aficionado 70    es socio y llego a las 18:40 horas.
        El aficionado 74    es socio y llego a las 18:54 horas.
        El aficionado 78    es socio y llego a las 18:33 horas.
        El aficionado 43 no es socio y llego a las 18:43 horas.
        El aficionado 49 no es socio y llego a las 18:29 horas.
        El aficionado 53 no es socio y llego a las 18:49 horas.
        El aficionado 57 no es socio y llego a las 18:45 horas.
        El aficionado 63 no es socio y llego a las 18:46 horas.
        El aficionado 69 no es socio y llego a las 18:48 horas.
        El aficionado 75 no es socio y llego a las 18:54 horas.
```

Por último con la **H** se eliminará un aficionado indicado por su ID (que se pide desde consola). Mostrará el árbol resultante tras la eliminación de dicho aficionado.

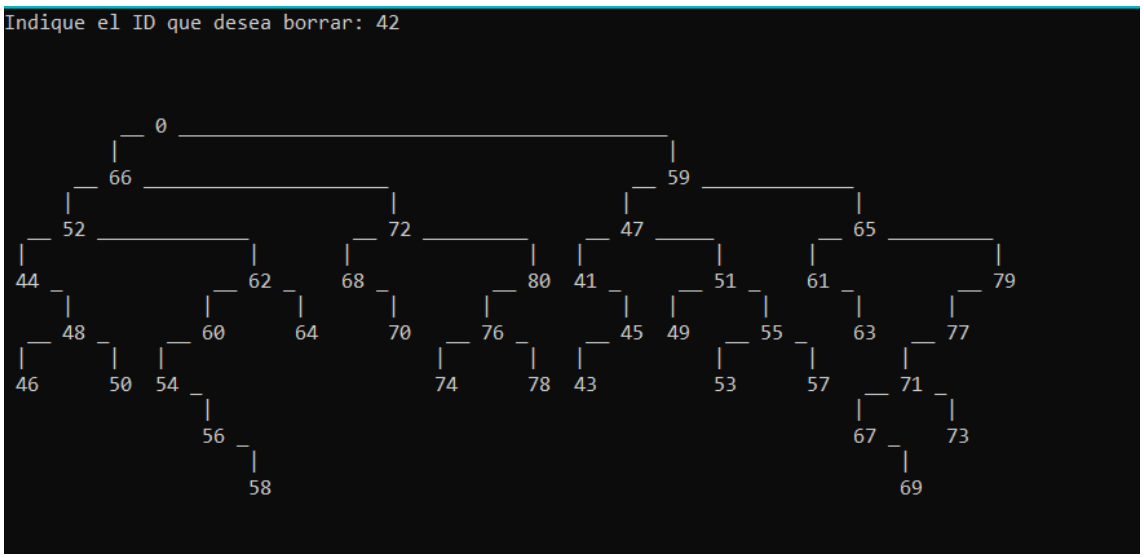
Siendo el árbol inicial el siguiente:



Si queremos eliminar el 75 quedaría así:



Si ahora deseamos eliminar el 42:



Por último, para salir del programa pulsamos **S**.

2. Detalles y justificación de la implementación:

2.1. Especificación concreta de la interfaz de los TAD´s implementados

2.1.1. TAD`s creados y definición operaciones

espec PILA[AFICIONADO]

usa BOOLEANOS

parámetro formal

géneros aficionado

fparámetro

géneros pila

operaciones

{Generadoras}

pvacía: →pila {crea una pila vacía}

insertar: aficionado pila → pila

{Modificadora}

parcial **extraer**: pila → pila

{Observadoras}

contar: pila → natural

vacía?: pila → bool

ecuaciones de definitud

vacía?(p) = F ⇒ Def(extraer(p))

fespec

espec COLA[AFICIONADO]

usa BOOLEANOS

parámetro formal

géneros aficionado

fparámetro

géneros cola

operaciones

{Generadoras}

cvacía: ⟶ cola {crea una cola vacía}

insertar: aficionado cola ⟶ cola {insertar un aficionado en la cola}

{Modificadora}

parcial **eliminar**: cola ⟶ cola

{Observadoras}

contar: cola ⟶ natural

vacía?: cola ⟶ bool

var c: cola

ecuaciones de definitud

vacía?(c) = F ⇒ Def(eliminar(c))

fespec

espec LISTA[AFICIONADO]

usa LISTA[AFICIONADO], NATURALES2

operaciones

{Ver un dato en una posición, insertar, modificar o borrar}

parcial **_ [_]**: lista natural → aficionado

parcial **insertar**: aficionado lista natural → lista

parcial **modificar**: aficionado lista natural → lista

parcial **borrar**: lista natural → lista

{Ver si un dato está en la lista, y en qué posición se encuentra}

está?: aficionado lista → bool

buscar: aficionado lista → natural

var

n : natural

afi : aficionado

l : lista

ecuaciones de definitud

(1 ≤ i ≤ long(l)) = T ⇒ Def(l[i])

(1 ≤ i ≤ long(l)) = T ⇒ Def(modificar(afi,l,i))

(1 ≤ i ≤ long(l)) = T ⇒ Def(borrar(l,i))

(1 ≤ i ≤ long(l)+1) = T ⇒ Def(insertar(afi,l,i))

fespec

Especificación de la segunda parte de la PECL:

Especificación. Árboles binarios

espec ÁRBOLES_BINARIOS+[AFICIONADO]

usa ÁRBOLES_BINARIOS[AFICIONADO], LISTAS[AFICIONADO]

operaciones

inorden: a_bin → lista {recorre en inorden}

var

afi: aficionado

i, d: a_bin

ecuaciones

inorden(Δ) = []

inorden(i•afi•d) = inorden(i) ++ [afi] ++ inorden(d)

fespec

espec ÁRBOLES_BINARIOS[AFICIONADO]

usa NATURALES2, BOOLEANOS

{NATURALES2 tiene operaciones para comparar números, como max, min, ≤, etc.}

parametro formal

generos aficionado

fparametro

generos a_bin {árbol_binario}

operaciones

$\Delta: \rightarrow a\_bin$ *{Generadoras libres}*

$\_\bullet\_\bullet\_: a\_bin$ aficionado $a\_bin \rightarrow a\_bin$

parcial raíz: a_bin → aficionado

parcial izq: a_bin → a_bin

parcial der: a_bin → a_bin

vacío?: a_bin → bool

parcial altura: a_bin → natural

var

afi: aficionado

a, i, d: a_bin

ecuaciones de definitud

vacía?(a) = F → Def(raíz(a))

vacía?(a) = F → Def(izq(a))

vacía?(a) = F → Def(der(a))

vacía?(a) = F → Def(altura(a))

ecuaciones

…

fespec

ÁRBOLES_BÚSQUEDA[ELEMENTO≤] {elem. con orden}

usa ÁRBOLES_BINARIOS[ELEMENTO]

parametro formal

generos elemento

operaciones { todas son "op: elemento elemento ® bool" }

_≤_

_<_

_>_

_≥ _

_=_

var

x, y: elemento

ecuaciones

$x = y \leftrightarrow (x \leq y) \wedge (y \leq x)$

fparametro

operaciones

insert : elemento a_bin → a_bin{inserta ordenadamente}

está? : elemento a_bin → bool {¿está el dato en el árbol?}

var

x, y: elemento

i, d: a_bin

ecuaciones

insert(x,Δ) = Δ•x•Δ

(y ≤ x) → insert(y, i•x•d) = insert(y, i)•x•d

(y > x) → insert(y, i•x•d) = i•x•insert(y, d)

está?(x,Δ) = F

(y < x) → está?(y, i•x•d) = está?(y, i)

(y = x) → está?(y, i•x•d) = T

(y > x) → está?(y, i•x•d) = está?(y, d)

fespec

Especificación árboles AVL

tipos

nodo_avl =**reg**

altura: natural

valor: elemento

izq: a_avl

der: a_avl

freg

a_avl = **puntero a** nodo_avl

ftipos

2.2. Solución adoptada: descripción de las dificultades encontradas.

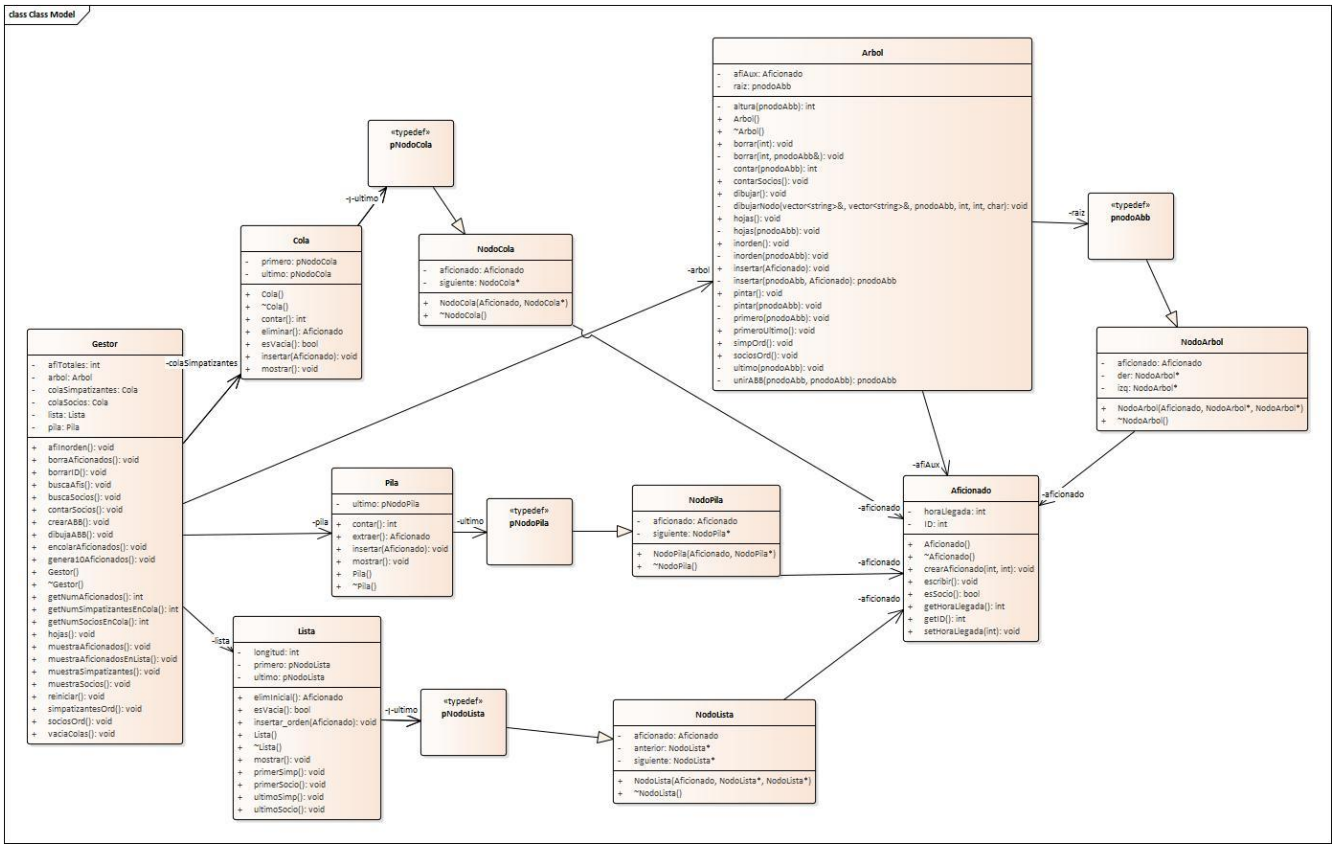
A la hora de realizar la práctica no hemos encontrado muchas dificultades más allá de errores de sintaxis exceptuando el apartado de lista.

Nosotros optamos por realizar una **lista doblemente** enlazada porque para añadir directamente en orden resulta menos costosa. Cuando un elemento es socio la recorrerá desde el primero hasta el último y cuando el elemento no sea socio será desde el último hasta el primero, todo ello se hará con un puntero auxiliar aux. Dentro de la función insertar_orden a veces cuando hay muchos elementos nos ocurre un fallo y es que al ejecutarla nos cierra el terminal, probando con el debugger encontramos que nos da fallo en la línea 32 con la instrucción: aux->anterior->siguiente = nuevo; Y comprobando los apuntes vemos que en la función insertar_orden de los apuntes se usa la misma instrucción y no hemos logrado averiguar a qué se debe este error.

En la segunda parte de la práctica solo encontramos una dificultad y era que a la hora de eliminar por ID no lo llegaba a borrar realmente, por ello nos percatamos tras muchas pruebas de que al pasarle el valor del nodo en vez del puntero nos funcionaba correctamente y sacamos la conclusión de que al pasarlo como puntero lo que haremos más adelante será comparar la dirección de memoria del ID con la dirección de memoria del ID del elemento del puntero y es por ello que de esa forma nunca se eliminaría un nodo.

2.3. Diseño de la relación entre las clases de los TAD implementados.

2.3.1. Diagrama UML.



2.3.2 Explicación de los métodos más destacados.

- *genera10Elementos():*

  Genera con el ID y la hora de llegada aleatorios (ID entre 1 y 10 y la hora entre 0 y 59). Cada vez que se use la opción 0 el id tendrá un valor aleatorio entre 1 y 10, 2 y 20, 3 y 30 y así sucesivamente.

```
void Gestor::genera10Aficionados()
{
    srand(time(NULL));
    int id, num[10], h;
    Aficionado afi;
    for(int i = 0; i < 10; i++) {
        do {
            id = 1 + rand() % 10;
        } while(repeticion(id, num));
        num[i] = id;
        h = rand() % 60;
        afi.CrearAficionado(id + afiTotales, h);
        pila.insertar(afi);
    }
    afiTotales=afiTotales+10;
}
```

- *muestraElementos():*

  Tiene como objetivo mostrar los 10 elementos generados aleatoriamente que se encuentran en la pila. Por eso, llama a pila.mostrar() que sigue mostrando los elementos de la pila hasta que

último sea nulo, es decir, hasta que la pila esté vacía y no haya más elementos que mostrar.

```
void Pila::mostrar()
{
    pNodoPila aux = ultimo;
    while(aux) {
        aux->aficionado.Escribir();
        aux = aux->siguiente;
    }
    cout << endl;
}
```

- *encolarElementos()*:

  Introduce los elementos en las colas. Comprueba uno a uno si es socio y si se cumple, lo introduce en colaSocios, sino, en colaSimpatizantes.

```
void Gestor::encolarAficionados()
{
    Aficionado afi;
    int nAficionados = getNumAficionados();
    for(int i = 0; i < nAficionados; i++) {
        afi = pila.extraer();
        if(afi.esSocio())
            colaSocios.insertar(afi);
        else
            colaSimpatizantes.insertar(afi);
    }
}
```

```
void Cola::insertar(Aficionado afi)
{
    pNodoCola nuevo;
    nuevo = new NodoCola(afi);
    if(ultimo)
        ultimo->siguiente = nuevo;
    ultimo = nuevo;
    if(!primero)
        primero = nuevo;
}
```

- *muestraSocios()*:

  Muestra los elementos de colaSocios con el método colaSocios.mostrar(). Este los muestra hasta que primero sea nulo, es decir, hasta que la cola esté vacía.

```
void Cola::mostrar()
{
    pNodoCola aux=primero;
    while(aux){
        aux->aficionado.Escribir();
        aux = aux->siguiente;
        }
        cout << endl;
}
```

- *muestraSimpatizantes()*:

  Mismo funcionamiento que muestraSocios() pero con colaSimpatizantes.

- *muestraElementosEnLista()*:

```
void Gestor::muestraAficionadosEnLista()
{
    while(!colaSocios.esVacia()) {
        lista.insertar_orden(colaSocios.eliminar());
    }
    while(!colaSimpatizantes.esVacia()) {
        lista.insertar_orden(colaSimpatizantes.eliminar());
    }
    lista.mostrar();
}
```

- *insertar_orden(): Con este método almacenaremos los elementos de forma ordenada en función de la hora de llegada y priorizandolos si son socios*

```
void Lista::insertar_orden(Aficionado afi)
{
    pNodoLista nuevo, aux;
    nuevo = new NodoLista(afi);
    if(esVacia()) {
        primero = nuevo;
        ultimo = nuevo;
    } else if(afi.esSocio()) {
        if(afi.horaLlegada < primero->aficionado.horaLlegada) {
            nuevo->siguiente = primero;
            primero->anterior = nuevo;
            primero = nuevo;
        } else {
            aux = primero;
            while(aux != NULL && aux->aficionado.horaLlegada < afi.horaLlegada && aux->aficionado.esSocio())
                aux = aux->siguiente;
            }
            if(aux == NULL) {
                ultimo->siguiente = nuevo;
                nuevo->anterior = ultimo;
                ultimo = nuevo;
            } else {
                nuevo->siguiente = aux;
                aux->anterior->siguiente = nuevo;
                nuevo->anterior = aux->anterior;
                aux->anterior = nuevo;
            }
        }
    } else {

        if(afi.horaLlegada > ultimo->aficionado.horaLlegada || ultimo->aficionado.esSocio()) {
            nuevo->anterior = ultimo;
            ultimo->siguiente = nuevo;
            ultimo = nuevo;
        } else {
            aux = ultimo;
            while(aux->aficionado.horaLlegada > afi.horaLlegada && !aux->aficionado.esSocio()) {
                aux = aux->anterior;
            }
            nuevo->anterior = aux;
            aux->siguiente->anterior = nuevo;
            nuevo->siguiente = aux->siguiente;
            aux->siguiente = nuevo;
        }
    }

    longitud = longitud + 1;
}
```

- *buscaSocios(): En este método ejecutaremos los siguientes métodos de la clase lista:*
  - *primerSocio(): Muestra por pantalla el primer socio de la lista*
  - *ultimoSocio():Muestra por pantalla el último socio de la lista*

- *primerSimp():Muestra por pantalla el primer simpatizante de la lista*
- *ultimoSimp():Muestra por pantalla el último simpatizante de la lista*

```
void Lista::primerSocio()
{
    cout << "El primer socio es: " << setw(20);
    primero->aficionado.Escribir();
}
void Lista::ultimoSocio()
{
    pNodoLista aux = primero;
    while(aux->aficionado.esSocio()) {
        aux = aux->siguiente;
    }
    cout << "El último socio es: " << setw(20);
    aux->anterior->aficionado.Escribir();
}
void Lista::primerSimp()
{
    pNodoLista aux = primero;
    while(aux->aficionado.esSocio()) {
        aux = aux->siguiente;
    }
    cout << "El primer simpatizante es: " << setw(10);
    aux->aficionado.Escribir();
}
void Lista::ultimoSimp()
{
    cout << "El último simpatizante es: " << setw(10);
    ultimo->aficionado.Escribir();
}
```

Segunda parte de la PECL (árboles binarios):

- *borrarID():* este procedimiento se encarga de eliminar un nodo en función de su ID que es pasado como parámetro, si el id es de socio llamará a otra función con el mismo nombre, pasándole como parámetro la rama izquierda y en el caso contrario la derecha. En este procedimiento realizaremos llamadas recursivas para recorrer el ABB hasta encontrar el nodo correspondiente, dirigiéndonos a la rama de la izquierda cuando el ID sea menor que el nodo y a la rama derecha en caso contrario. Una vez encontremos el nodo con el mismo ID crearemos un puntero auxiliar que apuntará al nodo, obtendremos el nuevo valor del nodo con la función unirABB y por último borraremos el valor anterior.

```
void Arbol::borrar(int ID)
{
    if(ID % 2 == 0) {
        borrar(ID, raiz->izq);
    } else {
        borrar(ID, raiz->der);
    }
}
void Arbol::borrar(int ID, pnodoAbb& nodo)
{
    if(nodo != NULL) {
        if(ID < nodo->aficionado.getID()) {
            borrar(ID, nodo->izq);
        } else if(ID > nodo->aficionado.getID()) {
            borrar(ID, nodo->der);
        } else {
            pnodoAbb paux = nodo;
            nodo = unirABB(nodo->izq, nodo->der);
            delete paux;
        }
    }
}
```

- *unirABB():* Con unirABB lo que haremos será reconstruir el ABB tras la eliminación de un nodo. Le pasaremos como parámetro el nodo izquierdo y el derecho del nodo a eliminar, si el nodo solo tiene un hijo, directamente devolverá ese hijo como valor para el nodo a eliminar y sino buscaremos de forma recursiva el centro del árbol llamando a la función hasta que izquierda o derecha sean nulos y por último en este caso giraremos el árbol haciendo que la derecha de la izquierda tome el valor del centro y el de la izquierda de la derecha tome el valor de la derecha y devolveremos la derecha.

```
pnodoAbb Arbol::unirABB(pnodoAbb izq, pnodoAbb der)
{
    if(izq == NULL) {
        return der;
    }
    if(der == NULL) {
        return izq;
    }
    pnodoAbb centro = unirABB(izq->der, der->izq);
    izq->der = centro;
    der->izq = izq;
    return der;
}
```

- *primeroUltimo():* Con este procedimiento obtendremos el primer elemento, último socio, primer simpatizante y último elemento en acceder al estadio. Primero, empezaremos obtendremos el primer elemento con la función primero que se encargará de recorrer el ABB inorden, almacenando el primero en la variable afiAux, a medida que encuentre un elemento lo comparará con afiAux y si dicho elemento tiene una hora de llegada menor pasará a ser el valor de afiAux. Si dicho elemento no es socio, el primer elemento será igual al primer simpatizante por lo que nos ahorraríamos tener que volver a ejecutar la función primero, pero si es el caso contrario, volveremos a ejecutar la función primero pero con la rama de la derecha y almacenaremos afiAux en primerSimp. Para obtener el último elemento lo que haremos será usar la función último que funciona igual que la de primero pero en vez de almacenar el primero, almacenará el último en afiAux. Si el último elemento es socio, el último socio será el mismo que el último elemento y ocurrirá lo mismo que con el primer elemento y simpatizante.

```
void Arbol::primeroUltimo()
{
    Aficionado primerAfi, ultimoSocio, primerSimp, ultimoAfi;
    afiAux.crearAficionado(0, 60);
    primero(raiz);
    primerAfi = afiAux;
    if(afiAux.esSocio()) {  //Si el aficionado no es socio, primerAfi será el mismo que primerSimp
        afiAux.setHoraLlegada(60);
        primero(raiz->der);
    }
    primerSimp = afiAux;
    afiAux.setHoraLlegada(0);
    ultimo(raiz);
    ultimoAfi = afiAux;
    if(!afiAux.esSocio()) { //Si el aficionado es socio, ultimoSocio será el mismo que ultimoSocio
        afiAux.setHoraLlegada(0);
        ultimo(raiz->izq);
    }
    ultimoSocio = afiAux;
    cout << "Primer aficionado en acceder al estadio: " << setw(10);
    primerAfi.escribir();
    cout << "Ultimo socio en acceder al estadio: " << setw(20);
    ultimoSocio.escribir();
    cout << "Primer simpatizante en acceder al estadio: " << setw(10);
    primerSimp.escribir();
    cout << "Ultimo aficionado en acceder al estadio: " << setw(10);
    ultimoAfi.escribir();
}
```

2.4. Explicación del comportamiento del programa.

El programa simula cómo se controla el acceso de unos elementos a un evento deportivo que empieza a las 19:00. En el menú podemos visualizar en todo momento el número de elementos en la pila y en las colas y debajo las opciones que se pueden realizar con el programa.

Primero, se generan 10 elementos pulsando la opción **0**. Los elementos tienen una hora de llegada (entre las 18:00 y las 18:59) y un identificador único. Ambos son generados aleatoriamente. En el caso del identificador, la primera vez que use esta opción será un número entre el 1 y el 10, la segunda entre el 11 y el 20, y así sucesivamente. Además, serán socios o simpatizantes según su identificador. (Si es par será socio y sino simpatizante).

Si pulsamos **1**, mostrará todos los elementos que hay en la pila.

Pulsando **2**, borrará todos los elementos almacenados en la pila.

Al pulsar la opción **3**, se almacenarán los elementos en dos colas. Dependiendo de si son socios o no se insertarán en la cola de los socios o de los simpatizantes.

Con el **4** mostrará la cola de los elementos socios y con el 5 la cola de los simpatizantes. Si se pulsa la opción 6 se borrarán todos los elementos almacenados en ambas colas.

La opción **7** extrae los elementos de ambas colas y los almacena en una lista de forma ordenada. Primero, se extraerán los elementos de la cola de los socios y se ordenarán en función de la hora de llegada de cada socio. Es decir, primero los que han llegado más pronto. Una vez hecho esto, se extraen los elementos simpatizantes y también se ordenan según su hora de llegada.

Pulsando **8** buscará en la lista cuatro elementos: el primer elemento en acceder al estadio, el último socio en acceder al estadio y el último elemento en acceder al estadio.

Si pulsamos **9** se reiniciará el programa al estado inicial.

Entonces se comienza con la segunda parte de la práctica

Pulsando **X** se creará el ABB cuya raíz será un elemento socio ficticio (cuyo ID será 0) y su subárbol izquierdo estará formado por los elementos que son socios ordenados por sus IDs en el subárbol izquierdo del nodo raíz y los que son simpatizantes en el subárbol derecho, también ordenados por sus IDs.

La opción **A** dibuja el árbol de elementos en la consola.

Con la opción **B** muestra los socios ordenados por su ID de menor a mayor sin contar el elemento ficticio. Mientras que con la opción **C** se muestran los simpatizantes también ordenados por sus IDs de menor a mayor.

La opción **D** muestra los datos de todos los elementos recorriendo el ABB en inorden es decir se recorre en inorden la rama izquierda, luego se visita la raíz, y después se recorre en inorden la rama derecha.

Pulsando **E** busca en el ABB y muestra el primer elemento en acceder al estadio, el último socio en acceder al estadio, el primer simpatizante en acceder al estadio y el último elemento en acceder al estadio.

Con la opción **F** recorre el ABB en busca de elementos socios, es decir, con IDs pares.

Pulsando **G** muestra los elementos que se encuentran almacenados en un nodo hoja comprobando que no hay subárbol izquierdo ni derecho.

Con la opción **H**, a la hora de borrar un nodo elemento hay distintos casos. Si el nodo elemento es una hoja se elimina directamente. Como hemos visto en el caso del elemento con el ID 75. Si el nodo eliminado solo tiene un hijo, conectamos el padre del nodo eliminado con ese hijo. Si tiene dos hijos el nodo eliminado puede reemplazarse por uno de estos elementos:

- El mayor ID de la rama izquierda (predecesor en inorden)
- El menor ID de la rama derecha (sucesor en inorden)

Para mantener el árbol como árbol de búsqueda, **el nodo borrado se sustituirá por el elemento mayor del hijo izquierdo**.

Y por último si pulsamos **S** saldrá del programa.

2.5 Bibliografía

Genera números aleatorios:

https://es.stackoverflow.com/questions/6040/rellenar-un-array-con-n%C3%BAmeros-aleatorios-sin-repetirse-los-n%C3%BAmeros

Para trabajar en equipo con github desktop: https://www.youtube.com/watch?v=77W2JSL7-r8

Apuntes de teoría de la asignatura