Estructuras arborescentes: árboles binarios

- Algunos de los ejercicios contenidos en este documento se han de resolver en el Jutge (en la lista correspondiente del curso actual); aquí están señalados convenientemente
- En general, los ejercicios contenidos en este documento se presentan por orden de dificultad. Por ello, recomendamos resolverlos en el orden en el que aparecen. No se supervisarán los problemas del Jutge si antes no se han resuelto los ejercicios previos.

Continuamos con los ejercicios de uso de las clases especiales vistas en clase de teoría, en este caso los árboles binarios. Los árboles están implementados en la clase `BinTree`, que no es estándar. Su fichero `BinTree.hh` está en `/assig/pro2/inclusions`. Si queréis solamente consultar la especificación de las operaciones, tenéis el fichero `BinTree.pdf` en la carpeta de la sesión (`/assig/pro2/arbres`).

Si observáis el código de la clase `BinTree`, veréis que se trata de una clase genérica, gracias al uso de `template`. Este mecanismo impide la compilación separada, pues el compilador de C++ no permite compilar código que use clases "templatizadas" sin ver las instanciaciones de las mismas. Por tanto la habitual división entre los ficheros `.hh` (para incluir) y `.cc` (para compilar) que hemos visto en otras clases no es útil. Por este motivo todo el código de la clase se encuentra en `BinTree.hh`. Este mecanismo es excepcional y su uso fuera del contexto de los `template` se considera inadecuado.

En la carpeta de la sesión se muestran ejemplos resueltos con árboles de enteros que usan ficheros `BinTreeIOint.hh` y `BinTreeIOint.cc`, con operaciones de lectura y escritura para árboles de enteros. Notad que dichas operaciones no pueden ser genéricas, pues dependen directamente del tipo del contenido de las estructuras. También hay ejemplos de como usar `tar` para hacer entregas en el *Jutge* y como codificar ficheros `Makefile`.

En varios ejercicios del *Jutge* se usa otra versión de la clase `BinTree` que da la opción de usar `cin` y `cout` para leer y escribir árboles cuyos elementos sean de tipos determinados[1] usando varios formatos de lectura y escritura.

Como norma general, aseguraos de usar la versión de `BinTree` que se presenta en el fichero `.zip` del icono del gatito correspondiente a cada ejercicio.

[1] Ver tutorial en la web de PRO2

1.1. Ejemplos

Los programas `mida_arbre.cc` y `pert_arbre_int.cc` muestran funciones que implementan respectivamente el tamaño de un árbol binario y la búsqueda de un elemento en un árbol binario. Probadlos junto con los ficheros `BinTreeIOint.hh` y `BinTreeIOint.cc` y la versión de `BinTree.hh` de la carpeta `inclusions`. También os proporcionamos ficheros con datos para dichos programas.

1.2. Ejercicios

1.2.1. Suma de los valores de un árbol

Ejercicio del *Jutge* X23429 de la lista *BinTree (1)*

Aquí podéis ver como se usan los formatos `VISUALFORMAT` e `INLINEFORMAT` para leer árboles.

1.2.2. Evaluar expresiones sin variables

Ejercicio del *Jutge* X74885 de la lista *BinTree (1)*

1.2.3. Búsqueda en un árbol de búsqueda binaria

Ejercicio del *Jutge* X75537 de la lista *BinTree (1)*

1.2.4. Árbol invertido

Ejercicio del *Jutge* X80177 de la lista *BinTree (1)*

Aquí podéis ver como se usan los formatos `VISUALFORMAT` e `INLINEFORMAT` para escribir árboles.

1.2.5. Árbol máximo

Ejercicio del *Jutge* X47839 de la lista *BinTree (1)*

1.2.6. Modificar los elementos de un árbol

Ejercicio del *Jutge* X11338 de la lista *BinTree (1)*

Producid dos soluciones, una en la que se obtenga un árbol nuevo a partir del original y otra en la que las modificaciones se apliquen directamente sobre el original.

Escribid un programa que dado un árbol de pares de enteros le sume un valor k al segundo elemento de cada par.

El ejercicio del Jutge es la versión `void`. Es necesario implementar operaciones de lectura y escritura de `BinTree` de `ParInt` para poder probar vuestra solución, pero solo hay que enviar lo que se pide en el Jutge.

1.2.7. Podar un árbol

Ejercicio del *Jutge* X50235 de la lista *BinTree (1)*

Dado un árbol binario de enteros `a` sin elementos repetidos y un entero `x`, eliminar de `a` el valor `x` y todos sus sucesores.

1.2.8. Búsqueda en un `BinTree` de pares de enteros.

Ejercicio del *Jutge* X20646 de la lista *BinTree (2)*

1.2.9. Evaluar expresiones con variables.

Ejercicio del *Jutge* X59547 de la lista *BinTree (2)*

1.2.10. Suma de los valores de un árbol a profundidad par.

Ejercicio del *Jutge* X91811 de la lista *BinTree (2)*

1.2.11. Camino más largo de un árbol.

Ejercicio del *Jutge* X91713 de la lista *BinTree (2)*

1.2.12. Número de izquierdas y derechas en un árbol.

Ejercicio del *Jutge* X86374 de la lista *BinTree (2)*

1.2.13. Borra ficheros con una cierta extensión.

Ejercicio del *Jutge* X24232 de la lista *BinTree (2)*

1.2.14. Evaluar expresiones con `string`.

Ejercicio del *Jutge* X69228 de la lista *BinTree (2)*

1.2.15. Árbol de grados de desequilibrio.

Ejercicio del *Jutge* X42470 de la lista *BinTree (3)*

1.2.16. Árbol de alturas.

Ejercicio del *Jutge* X13384 de la lista *BinTree (3)*

1.2.17. Árbol de sumas.

Ejercicio del *Jutge* X63560 de la lista *BinTree (3)*

1.2.18. Árbol de tamaños.

Ejercicio del *Jutge* X56129 de la lista *BinTree (3)*

1.2.19. Árbol de ordenación.

Ejercicio del *Jutge* X92557 de la lista *BinTree (3)*

1.2.20. Reemplazar los valores de un nodo a profundidad par en un árbol por la suma de debajo.

Ejercicio del *Jutge* X68048 de la lista *BinTree (3)*