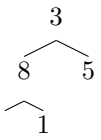
07 - Recorridos de árboles binarios (EVALUACIÓN)

Descripción



Extiende la clase `bintree` vista en clase (fichero `bintree.h`) para incorporar los siguientes métodos públicos:

- `vector<T> preorder() const`
- `vector<T> inorder() const`
- `vector<T> postorder() const`
- `vector<T> levels() const`

Estos métodos devuelven un `vector` con el recorrido en preorden, inorden, postorden y por niveles respectivamente. Todos ellos deben tener un coste $\in O(n)$, donde n es el número de elementos en el árbol binario. Además, como mínimo `preorder`, `inorder` y `postorder` se deben implementar de manera recursiva, quizá invocando a métodos privados recursivos `preorder_rec`, `inorder_rec` y `postorder_rec` que realizan el cálculo propiamente dicho.

Importante: A la hora de realizar el envío al juez, debes elegir el fichero `.cpp` junto con todos los ficheros cabecera `.h` que necesites, concretamente el fichero `bintree.h` modificado. Si esto os da algún problema, recomiendo combinar todo el código fuente en un único fichero `main.cpp` que contenga la definición de la clase `bintree.h` modificada además de la función `main` y demás funciones auxiliares que necesitéis.

Entrada

La entrada comenzará con una linea conteniendo un número natural N que indica la cantidad casos de prueba que vamos a considerar. Cada caso de prueba será una línea con números enteros separados por espacios conteniendo la descripción de un árbol binario. El valor 0 indicará un árbol vacío. Si el número es diferente de 0 se tratará de la raíz del árbol, que será seguida de la descripción del hijo izquierdo y luego de la descripción del hijo derecho. Por ejemplo, el árbol de la figura se describiría como:

$$3 \quad \underbrace{8\,0\,1\,0\,0}_{\text{hijo izquierdo}} \quad \underbrace{5\,0\,0}_{\text{hijo derecho}}$$

Salida

Por cada caso de prueba, la salida debe ser 4 líneas: el recorrido en preorden, el recorrido en inorden, el recorrido en postorden y el recorrido por niveles. Al mostar

cada uno de los recorridos se debe separar los elementos con exactamente un espacio, **pero no se espera un espacio después del último elemento**. Después de cada caso debéis imprimir una línea con dos símbolos de igual `==` para delimitar claramente la salida.

Ejemplo de entrada

```
2
5 0 0
3 8 0 1 0 0 5 0 0
```

Ejemplo de salida

```
5
5
5
5
==
3 8 1 5
8 1 3 5
1 8 5 3
3 8 5 1
==
```