Universidad de Costa Rica

Escuela de Ciencias de la Computación e Informática

CI-0113 Programación II

**Tarea Programada 4**

Documentación externa

Estudiantes:

* Sebastián Cruz Chavarría – B72458
* Roy Andrés Rojas Angulo – B76629

Prof. Edgar Casasola Murillo

I – 2018

Índice

Descripción del problema ……………………………………………………………………. 3

Descripción de la solución …………………………………………………………………….4

Diagrama de clases ……………………………………………………………………………6

Descripción del programa ……………………………………………………………………..7

Compilación del programa ……………………………………………………………………

Casos de prueba ………………………………………………………………………………

Formato de los casos de prueba …………………………………………………………….

Salida esperada y obtenida …………………………………………………………………..

Lista de archivos entregados …………………………………………………………………

Descripción del problema

En esta tarea programada se nos planteó como objetivo o problema programar un árbol binario equilibrado, más específicamente un árbol rojo-negro, además, se nos pidió que los pasos de inserción de pares ordenados en el árbol fueran visualizados mediante archivos de formato .svg. Para la solución de este problema fue necesario conocimiento sobre todos los temas vistos en clase, como lo son manejo de memoria, herencia, polimorfismo y plantillas. Este último es el que nos permite diseñar el árbol de manera tal que pueda recibir pares ordenados (llave, valor) de cualquier tipo, eso si la llave es un elemento comparable.

Primeramente, uno de los retos que surgieron para la resolución de esta tarea fue la implementación de un iterador que funcionará sobre el Arbol, la dificultad de este consiste en que se pueda mover entre padres e hijos directos y que no “brinque” a otras ramas.

Hubo también dificultad en la creación del algoritmo de equilibrio del Arbol, cómo se iban a detectar los casos en los que se debía aplicar rotaciones o cambios de color en los nodos y en qué orden.

Se tuvo que realizar ajustes en el diseño del SVG para la visualización del Arbol, ya que en algunas ocasiones ciertos nodos se sobreponían, imposibilitando comprender correctamente cómo se desplegaban las ramas.

Pero sin duda alguna el mayor problema fue poder instanciar el Arbol que se solicitaba en el archivo de texto o como argumento en la consola, ya que desde cualquiera de estas dos fuentes el tipo K de llave y V de valor se recibiría como una cadena de caracteres, la cual no se podría simplemente insertar en los campos correspondientes a K y V en los parámetros de la plantilla.

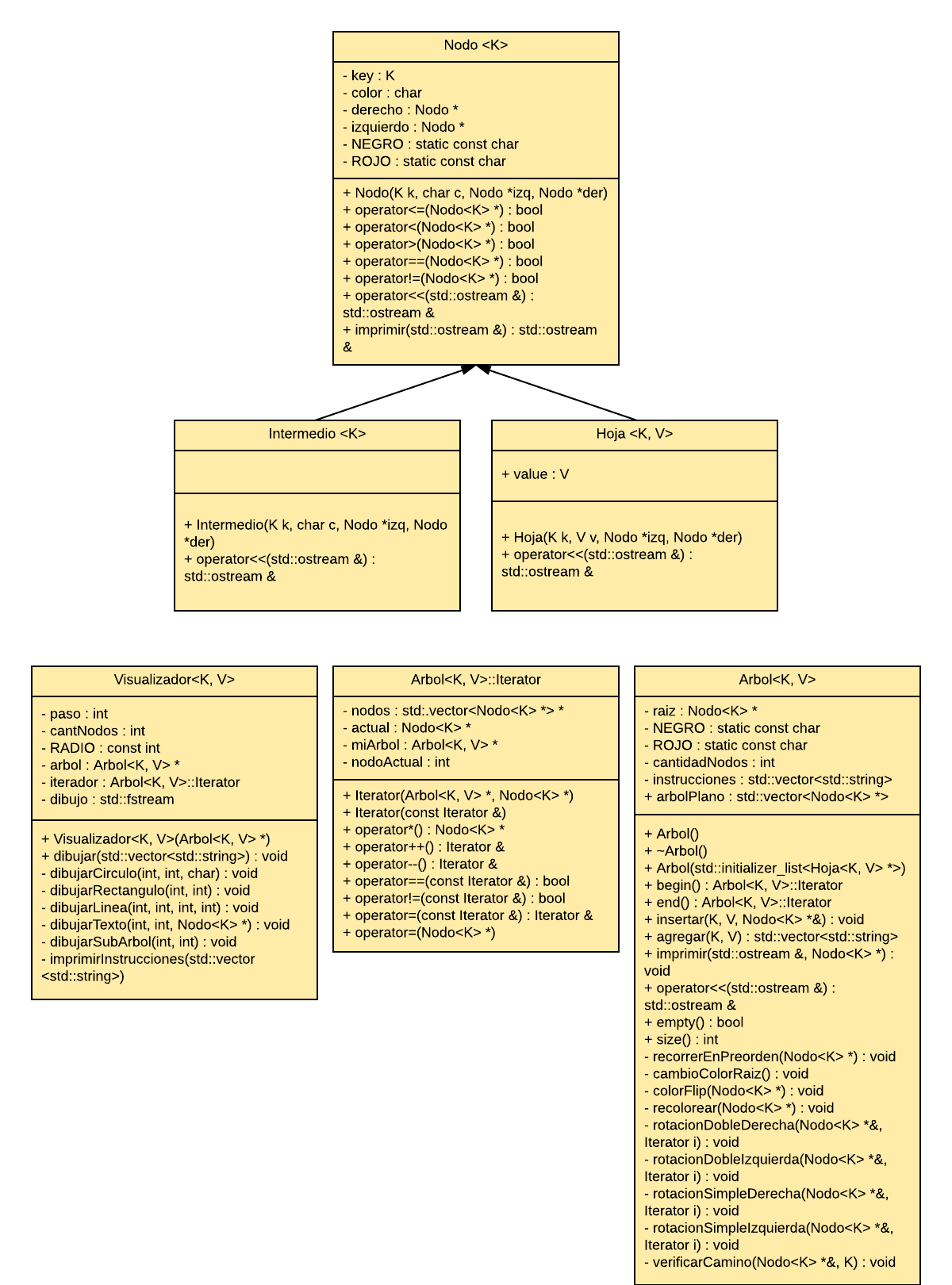
Descripción de la solución

Para la solución de este problema fue necesaria la implementación de cinco clases:

* En la clase Arbol se ubican todos los métodos necesarios para llevar a cabo su equilibrio, por supuesto también podemos encontrar las funciones con las cuales se permite agregar datos a este. Claramente es en este objeto donde convergen y se relacionan las demás clases, excepto tal vez Visualizador la cual más bien existe únicamente como amiga en Arbol para cumplir con sus funciones. Cabe destacar que la clase Iterator se encuentra anidada en Arobl, la cual había sido mencionada anteriormente debido a su forma de funcionar. Iterator es de suma importancia para hacer posible las rotaciones que requiere el Arbol en ciertos momentos.
* Existe la clase emplantillada abstracta Nodo, esta es base de las clases concretas Hoja e Intermedio. A grandes rasgos posee una llave K que puede ser de cualquier tipo (mientras ese tipo sea comparable), esta llave le da un “peso” al Nodo de tal manera que al insertar nodos al Arbol estos se agreguen de manera ordenada. Eso no implica equilibrio, sólo orden en la estructura para cumplir con la parte ordenada del árbol binario. Posee únicamente el método imprimir como virtual puro, pero este es suficiente para hacer de esta clase una de tipo abstracta, o polimórfica. También, implementa varias funciones de operadores relacionales que hereda a sus clases derivadas, dado que tanto un nodo Intermedio como una Hoja tienen un campo para su llave, se comparan por medio de esta variable.
* Intermedio, esta es una clase derivada de Nodo, la cual es también emplantillada, pero no padre de alguna otra. No agrega ningún atributo extra, a diferencia de Hoja. Forma parte vital de la estructura del Arbol, debido a la llave que contiene esta clase, sirve de mediadora para la búsqueda de algún dato en este, en el caso de que fuese necesario, así como para el ordenamiento del árbol.
* La clase Hoja derivada de Nodo alberga además de su llave un valor (V), el cual podemos relacionarlo como un dato. En su implementación del método imprimir se encarga de mostrar ambos, por su parte Intermedio sólo aporta la llave (K). Como su nombre lo indica, estos nodos son el final de sus ramas, y en el caso de esta tarea programada son los que contienen los datos del Arbol.
* Como última pero no menos importante se encuentra la clase Visualizador. Esta se encarga de elaborar un archivo SVG que contiene un dibujo del Arbol, y muestra los pasos que se llevan a cabo para insertar un dato u hoja. Por supuesto contiene todos los métodos necesarios para dibujar los Nodos, tanto Intermedios como Hojas, con sus correspondientes valores y colores.

El diseño de clases se implementó de esta forma dada la similitud entre las clases Intermedio y Hoja, siendo esta última la que contiene el campo para guardar el valor que representa, y una función de inserción de flujo donde mostrará su llave y su valor. Así, se aprovecharon las ventajas del polimorfismo, para tratar a todos los nodos como un Nodo de tipo K (Nodo<K>) genérico y poder manipular fácilmente los nodos del árbol en las operaciones que así lo requieran. En la figura 1 se aprecia mejor el diseño de la jerarquía de clases de Nodos, así como las otras clases de las cuales ya se ha hablado.

Diagrama de clases UML

Figura 1. Diagrama de clases UML.

Descripción del programa