

Laboratorio 5

Estructuras de Datos

Santiago Cardona 160004910

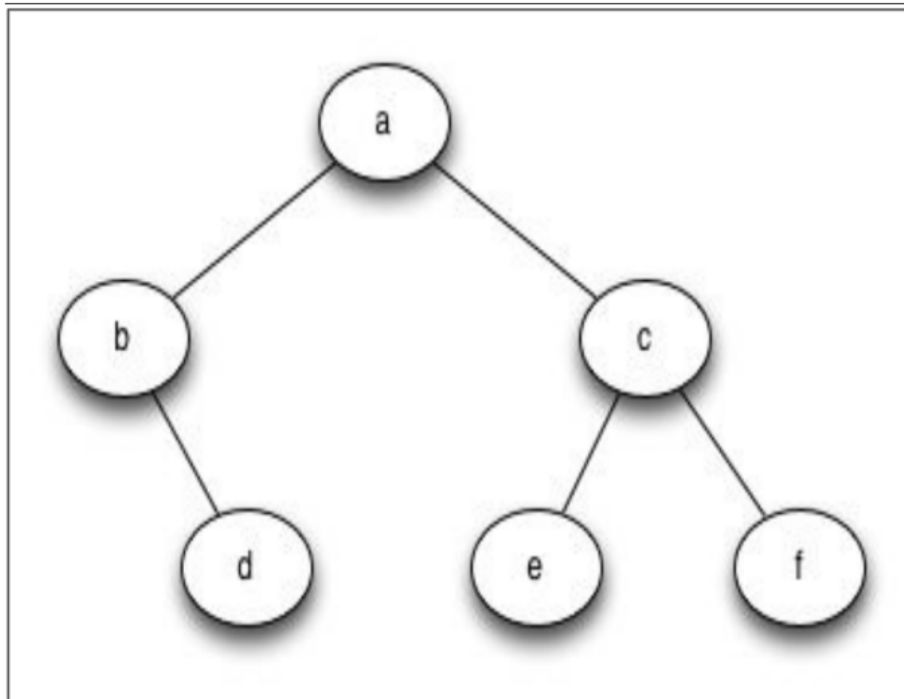
Juan Aristizabal 160004903

Noviembre 25, 2023

1. Punto 1

1.1 Enunciado

Escriba una función `crearArbol()` que devuelva un árbol usando la implementación de nodos y referencias y que corresponda al siguiente árbol:



1.2 Código fuente

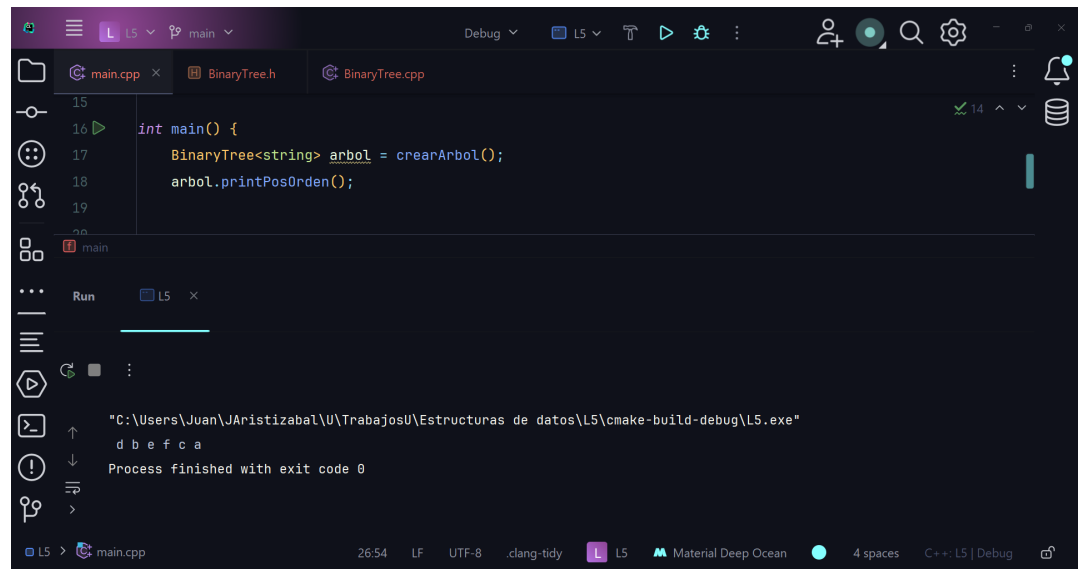
```
BinaryTree<string> crearArbol() {  
    BinaryTree<string> arbol("a");  
    arbol.getRoot()->insertLeft("b");  
    arbol.getRoot()->insertRight("c");  
    arbol.getRoot()->left->insertRight("d");  
    arbol.getRoot()->right->insertLeft("e");  
    arbol.getRoot()->right->insertRight("f");  
    return arbol;  
}
```

1.3 Descripción

La función `crearArbol` es una función que no recibe parámetros, pero si retorna un árbol binario de tipo `string`. Esta función realiza la creación de un árbol binario de tipo `string`, cuyo dato que almacena su nodo raíz es "a". Luego se crean dos nodos que almacenan los datos "b" y "c", que corresponden a los nodos izquierdo y derecho del nodo base "a" respectivamente. Seguidamente, se accede al nodo izquierdo de "a", es decir, "b", y se le crea un nodo hijo derecho a dicho nodo, el cual almacena el dato "d". Adicionalmente, se accede al nodo derecho de "a", es decir, "c", y se le crea un hijo izquierdo y otro derecho, que almacenan los datos "e" y "f" respectivamente. Finalmente, la función retorna el árbol binario de tipo `string` creado y usado.

Para su ejecución, se tuvo que definir un constructor copia de la clase `BinaryTree`, esto con el propósito de poder hacer la creación de un árbol binario en la función `main` e inmediatamente hacer el llamado a la función `crearArbol` la cual le proporciona al árbol creado, un árbol nuevo, del cual debe copiar su estructura.

1.4 Resultados



```
15
16 int main() {
17     BinaryTree<string> arbol = crearArbol();
18     arbol.printPosOrden();
19 }

Run

"C:\Users\Juan\JAriztizabal\U\Trabajos\Estructuras de datos\L5\cmake-build-debug\L5.exe"
d b e f c a
Process finished with exit code 0
```

2. Punto 2

2.1 Enunciado

Escriba un algoritmo que cuente la cantidad de nodos de un árbol binario.

2.2 Código fuente

```
template<class T>
int BinaryTree<T>::getCountNodes(NodeTree<T> *ptr) const {
    if (!ptr) return 0;
    return getCountNodes(ptr->left) + getCountNodes(ptr->right) + 1;
}

template<class T>
```

```
void BinaryTree<T>::getCountNodes() {
    std::cout << "La cantidad de nodos del arbol [" << root->data << "]"
es: "<< getCountNodes(this->root) << std::endl;
}
```

2.3 Descripción

La función `getCountNodes`, es un método miembro de la clase `BinaryTree`, es una función que está definida como template, `void`. Debido a que la recursividad facilita el recorrer un árbol, por esa razón se crea una función helper llamada de la misma manera que la principal y que recibe un puntero `NodeTree<T>`, ésta es la encargada de la recursividad. La función helper tiene un caso base en el que el puntero que recibe como parámetro es vacío o no apunta a nada; esto significa que se ha llegado a una hoja del árbol y debe retornar 0. También tiene otra línea en la cual retorna, la cual es el caso recursivo, primero enviando como parámetro el puntero izquierdo del nodo al cual apunta el puntero que se recibió en el llamado de la función, sumándole a esto, el llamado a la función, pero ahora con el puntero derecho y finalmente sumándole 1.

La función itera hasta encontrar los nodos hojas y luego se devuelve calculando la cantidad de nodos en el camino. Se le suma 1 porque la función va calculando la cantidad de nodos hijos, pero no tiene en cuenta al padre de estos, por lo cual es necesario adicionarle 1 para tenerlos en cuenta.

Luego en la función principal se hace el llamado a la función helper, enviándole como parámetro, la raíz del árbol y mostrando por pantalla su valor.

2.4 Resultados

The screenshot shows a C++ IDE with the following code in `main.cpp`:

```
14 }
15
16 int main() {
17     BinaryTree<string> arbol = crearArbol();
18     arbol.getCountNodes();
19 }
```

The output window shows the following message:

```
"C:\Users\Juan\JAriztizabal\U\TrabajosU\Estructuras de datos\L5\cmake-build-debug\L5.exe"
La cantidad de nodos del arbol [a] es: 6
Process finished with exit code 0
```

3. Punto 3

3.1 Enunciado

Escriba un algoritmo que cuente la cantidad de hojas de un árbol binario.

3.2 Código fuente

```
template<class T>
int BinaryTree<T>::getNumLeaves(NodeTree<T>* ptr) {
    if(!ptr) return 0;
    if (ptr->left == nullptr && ptr->right == nullptr) {
        // Es una hoja
        return 1;
    }
    return getNumLeaves(ptr->left) + getNumLeaves(ptr->right);
}

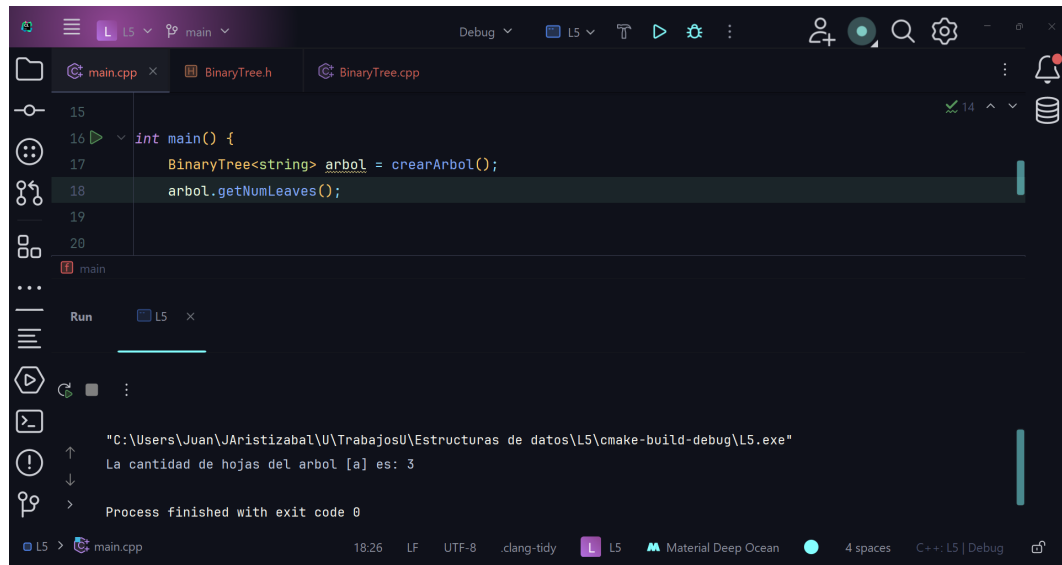
template<class T>
void BinaryTree<T>::getNumLeaves() {
    std::cout << "La cantidad de hojas del arbol [" <<
root->data << "]" es: " << getNumLeaves(root) <<
std::endl;
}
```

3.3 Descripción

La función `getNumLeaves` es un método miembro de la clase `BinaryTree`, está definida como `template, void`. De la misma forma que en el punto anterior, para este caso también se hace uso de una función helper la cual se encarga de la recursividad. La función helper recibe como parámetro un puntero `NodeTree<T>`; el caso base es cuando el puntero recibido no apunta a nada, en ese caso retorna 0; existe un segundo caso en el cual el puntero apunta a un nodo, pero los punteros izquierdo y derecho de ese nodo no apuntan a nada, es decir, sería una hoja; para este caso la función retorna 1, porque es exactamente lo que se requiere contabilizar, el número de hojas. Finalmente se retorna y se hace el llamado recursivo, primero con el puntero izquierdo del nodo al cual apunta el puntero recibido como parámetro y luego con el puntero derecho.

Luego en la función principal se hace el llamado a la función helper, enviándole como parámetro, la raíz del árbol y mostrando por pantalla su valor.

3.4 Resultados



4. Punto 4

4.1 Enunciado

Escriba un algoritmo que determine la altura de un árbol binario.

4.2 Código fuente

```
int max(int& a, int& b) {  
    return (a > b) ? a : b;  
}  
  
template<class T>  
int BinaryTree<T>::getTreeHeight(NodeTree<T> *ptr) const {  
    if (!ptr) return 0;  
    int leftHeight = getTreeHeight(ptr->left);  
    int rightHeight = getTreeHeight(ptr->right);  
    return max(leftHeight, rightHeight) + 1;  
}  
  
template<class T>  
void BinaryTree<T>::getTreeHeight() {  
    std::cout << "La altura del arbol [" << root->data << "] es:  
" << getTreeHeight(this->root) - 1 << std::endl;  
}
```

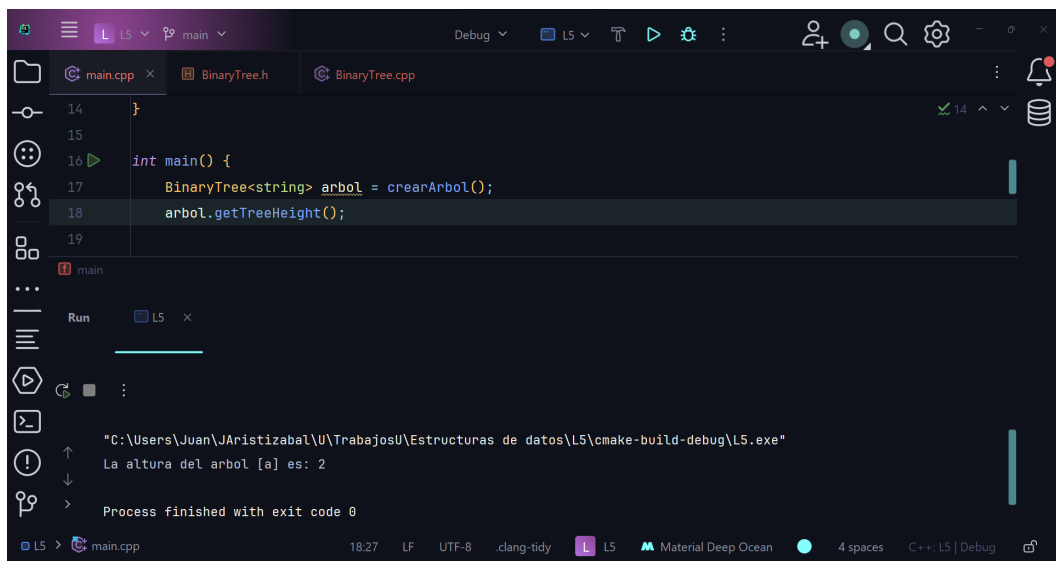
4.3 Descripción

La función `getTreeHeight` es un método miembro de la clase `BinaryTree`, está definida como `template`, `void`. De la misma forma que en los puntos anteriores, para este caso también se hace uso de una función helper la cual se encarga de la recursividad. La función helper recibe como parámetro un puntero `NodeTree<T>`; el caso base es cuando el puntero recibido no apunta a nada, en ese caso retorna 0. Para el caso recursivo, hace el llamado recursivo dos veces, pero en este caso se almacena el primero en una variable con el identificador `leftHeight` y el segundo en una con el identificador `rightHeight`; se le asignan como argumentos, para la primera llamada, el puntero izquierdo del nodo al cual apunta el puntero recibido como parámetro y para la segunda, el puntero

derecho del nodo al cual apunta el puntero recibido como parámetro. Finalmente se retorna y se hace el llamado a una función llamada max, definida en el mismo archivo, pero que no es miembro de la clase, la cual recibe dos números como parámetros y devuelve el número mayor. A esta función max se le envían las variables leftHeight y rightHeight; al valor que retorna la función max se le suma 1.

Luego en la función principal se hace el llamado a la función helper, enviándole como parámetro, la raíz del árbol y mostrando por pantalla su valor, menos 1, porque la función helper cuenta dos veces la arista al nodo raíz.

4.4 Resultados



```
14 }
15
16 int main() {
17     BinaryTree<string> arbol = crearArbol();
18     arbol.getTreeHeight();
19 }

Run L5 x

"C:\Users\Juan\JArizabal\U\Trabajos\Estructuras de datos\L5\cmake-build-debug\L5.exe"
La altura del arbol [a] es: 2
Process finished with exit code 0
```

5. Punto 5

5.1 Enunciado

Escriba una función que calcula el factor de balance de un árbol binario. Si este es llamado inicialmente con el puntero a raíz, debería determinar el factor de balance del árbol entero. Si este es llamado con un puntero a un subárbol, este debería determinar el factor de balance del subárbol.

5.2 Código fuente

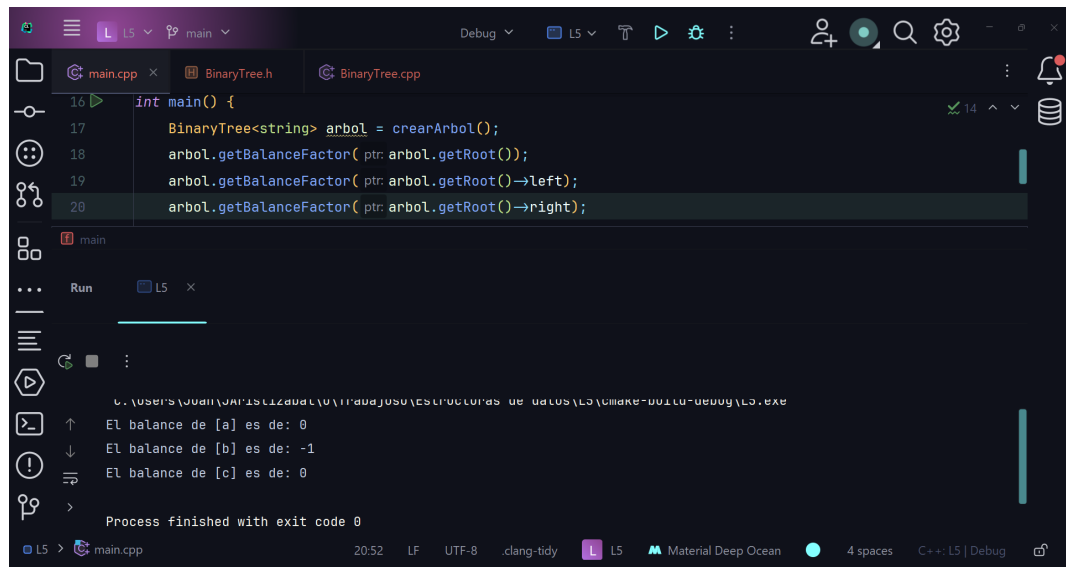
```
template<class T>
void BinaryTree<T>::getBalanceFactor(NodeTree<T> *ptr) {
    int leftHeight = getTreeHeight(ptr->left);
    int rightHeight = getTreeHeight(ptr->right);
    std::cout << "El balance de [" << ptr->data << "] es de: "
<< (leftHeight - rightHeight) << std::endl;
}
```

5.3 Descripción

La función getBalanceFactor es un método miembro de la clase BinaryTree, está definida como template, void. Esta función hace uso de la función helper llamada getTreeHeight, descrita anteriormente, la cual calcula la altura del árbol. En este

caso se llama dos veces, a la primera se le asigna como argumento el puntero izquierdo del nodo al cual está apuntando el puntero que recibe la función `getBalanceFactor` y el resultado se guarda en una variable con el identificador `leftHeight`; a la segunda se le asigna como argumento el puntero derecho del nodo al cual está apuntando el puntero que recibe la función `getBalanceFactor` y el resultado se guarda en una variable con el identificador `rightHeight`. Estos valores obtenidos corresponden a la altura del subárbol izquierdo y derecho, respectivamente, del nodo raíz "a". Finalmente se restan las variables `leftHeight` y `rightHeight` para encontrar el factor de balance del árbol, y se muestra por pantalla. Esta función puede calcular el factor de balance de cualquier árbol o subárbol que se le envíe como argumento.

5.4 Resultados



```
16 int main() {
17     BinaryTree<string> arbol = crearArbol();
18     arbol.getBalanceFactor(ptr: arbol.getRoot());
19     arbol.getBalanceFactor(ptr: arbol.getRoot()→left);
20     arbol.getBalanceFactor(ptr: arbol.getRoot()→right);
}
```

```
El balance de [a] es de: 0
El balance de [b] es de: -1
El balance de [c] es de: 0

Process finished with exit code 0
```