

ESTRUCTURAS DE DATOS

TIPOS ABSTRACTOS DE DATOS ARBORESCENTES

Implementando recorridos en anchura (*BFS*)

Manuel Montenegro Montes
Departamento de Sistemas Informáticos y Computación
Facultad de Informática – Universidad Complutense de Madrid

Tipos de recorridos

- Recorrido en profundidad

Depth First Search (DFS)

- Preorden
- Inorden
- Postorden

- Recorrido en anchura

Breadth First Search (BFS)



Nuevo método

```
template<class T>
class BinTree {
public:
    // ...
    void preorder() const;
    void inorder() const;
    void postorder() const;

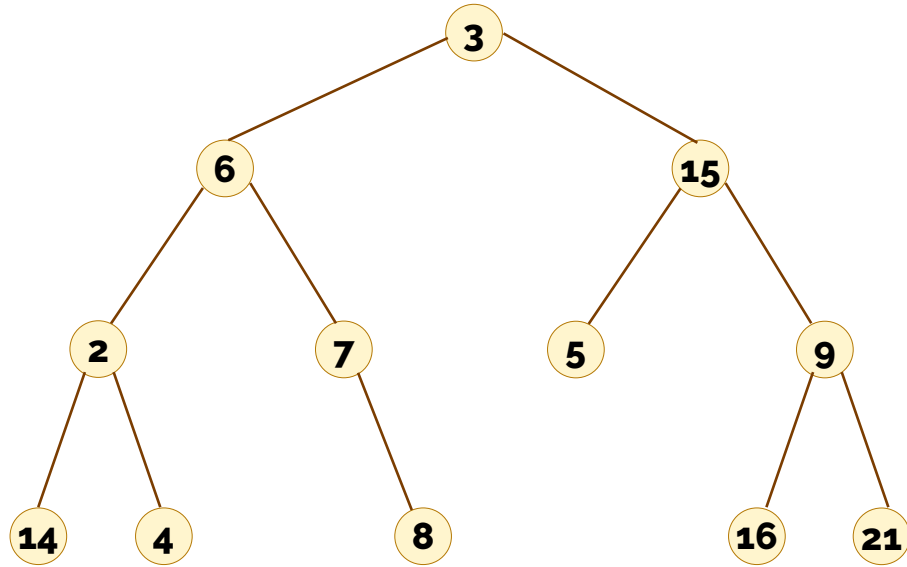
    void levelorder() const;

private:
    ...
};
```

- Implementamos el recorrido en profundidad mediante un nuevo método.



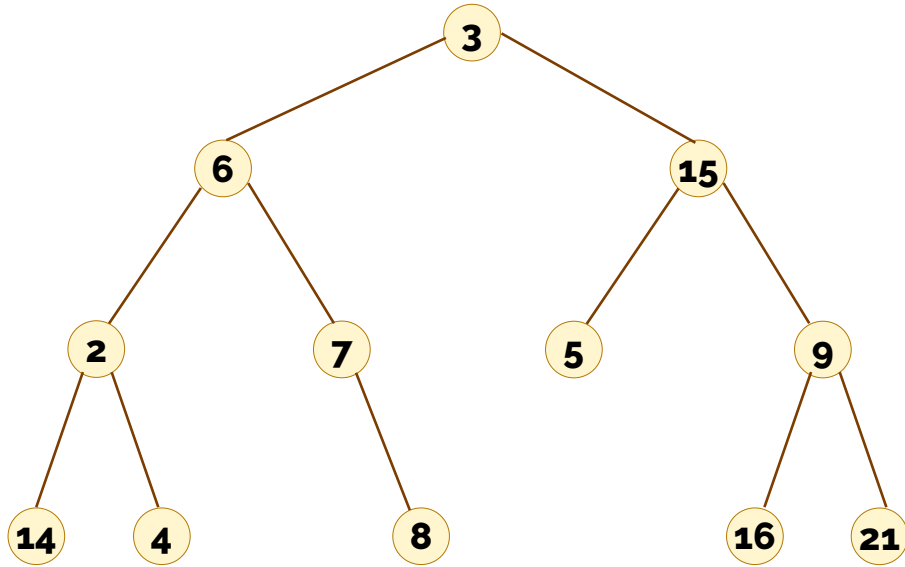
Algoritmo de recorrido en anchura



- Utilizaremos una **cola** que contiene punteros a nodos.
- Esta cola representa los nodos pendientes que nos quedan por visitar.

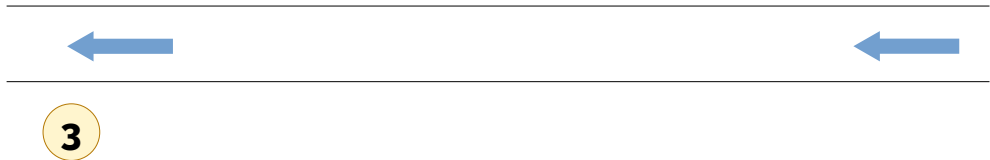
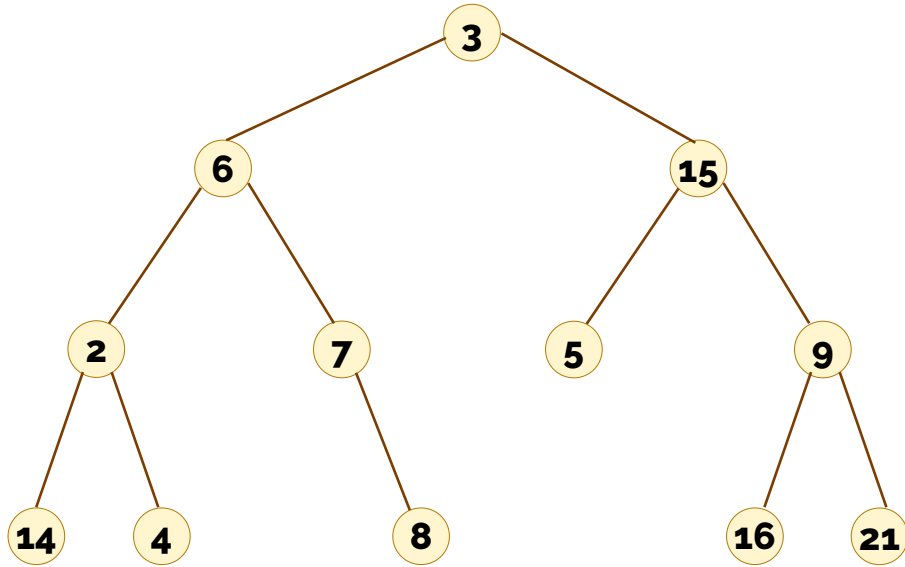
Algoritmo de recorrido en anchura

- Insertamos la raíz en la cola.
- Repetimos:

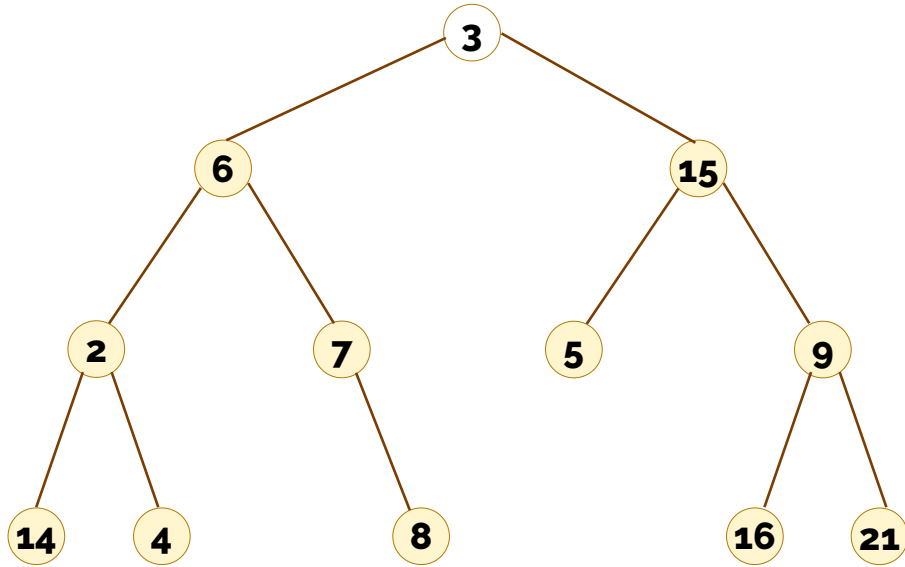


Algoritmo de recorrido en anchura

- Insertamos la raíz en la cola.
- Repetimos:
 - Sacamos nodo de la cola.

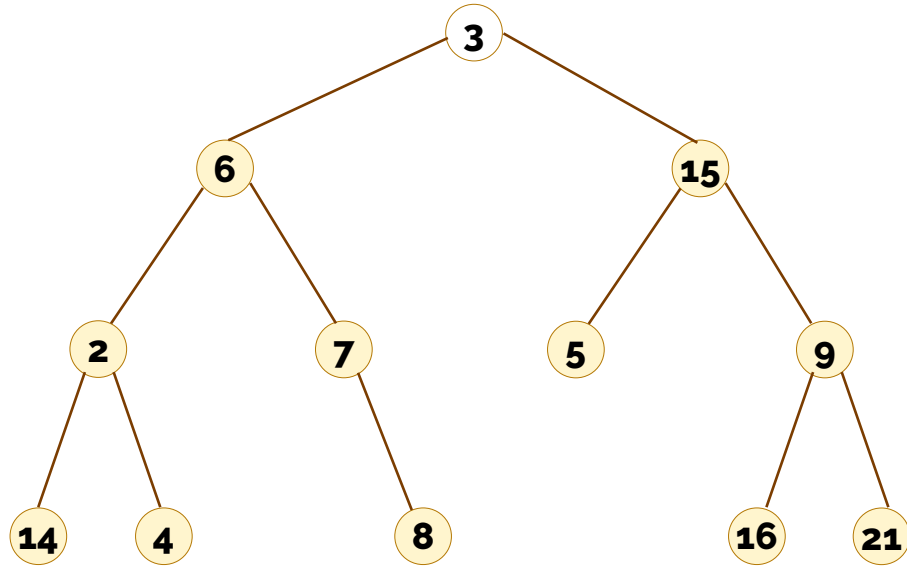


Algoritmo de recorrido en anchura



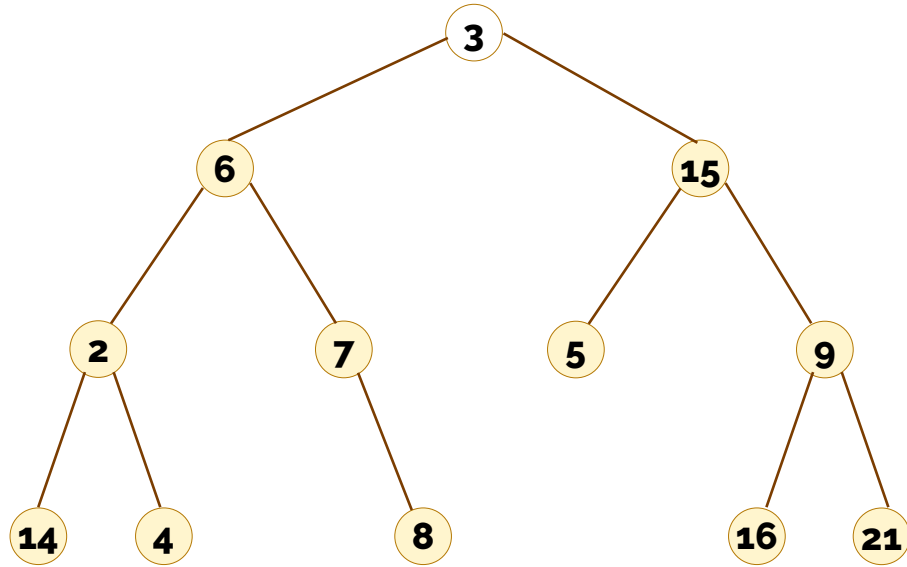
- Insertamos la raíz en la cola.
- Repetimos:
 - Sacamos nodo de la cola.
 - Visitamos ese nodo.

Algoritmo de recorrido en anchura

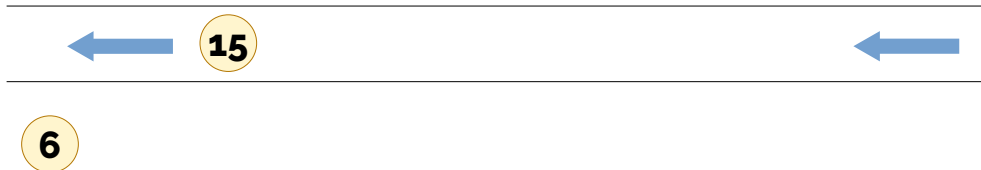


- Insertamos la raíz en la cola.
- Repetimos:
 - Sacamos nodo de la cola.
 - Visitamos ese nodo.
 - Insertamos sus hijos en la cola.

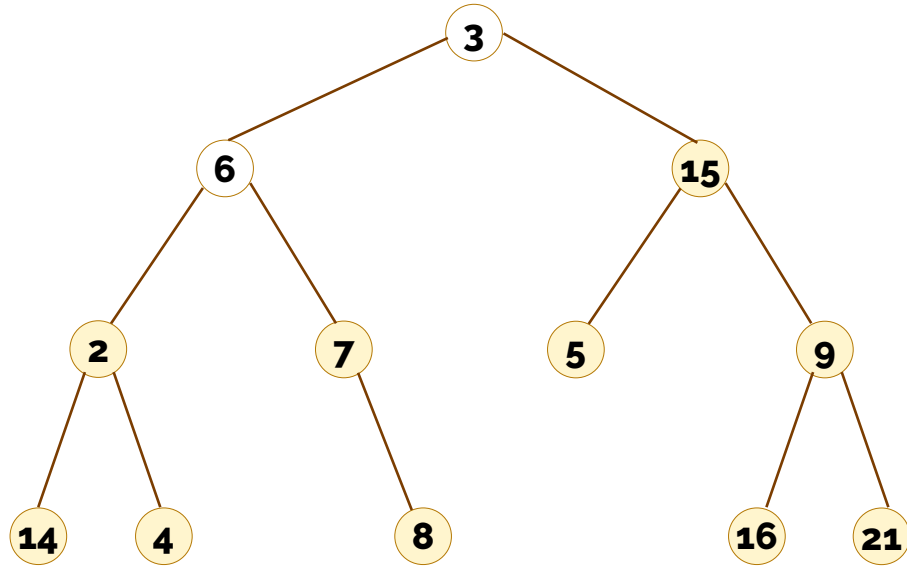
Algoritmo de recorrido en anchura



- Insertamos la raíz en la cola.
- Repetimos:
 - Sacamos nodo de la cola.
 - Visitamos ese nodo.
 - Insertamos sus hijos en la cola.

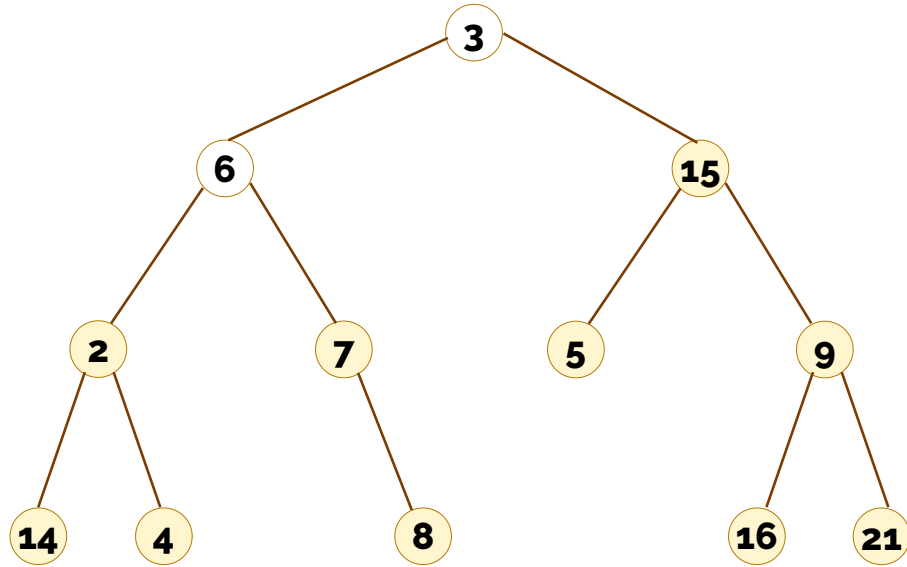


Algoritmo de recorrido en anchura



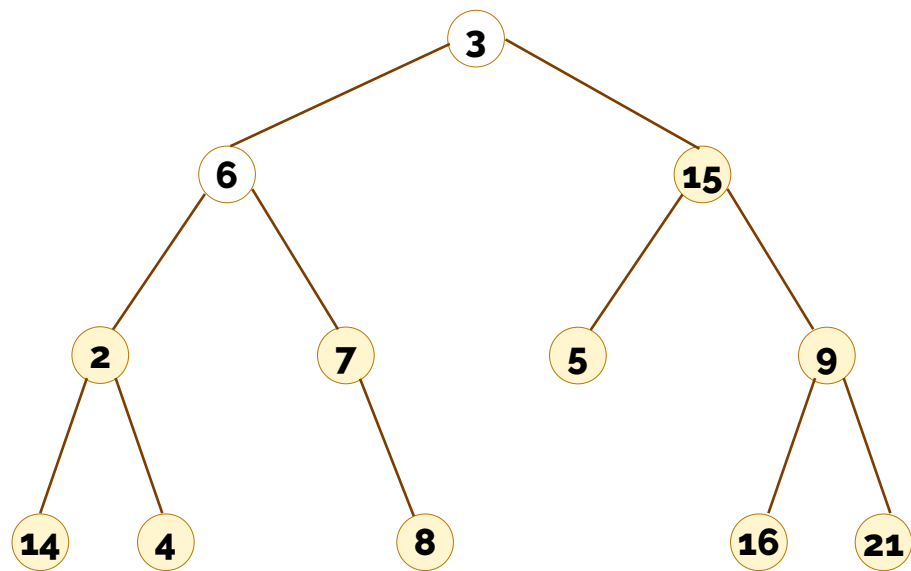
- Insertamos la raíz en la cola.
- Repetimos:
 - Sacamos nodo de la cola.
 - Visitamos ese nodo.
 - Insertamos sus hijos en la cola.

Algoritmo de recorrido en anchura

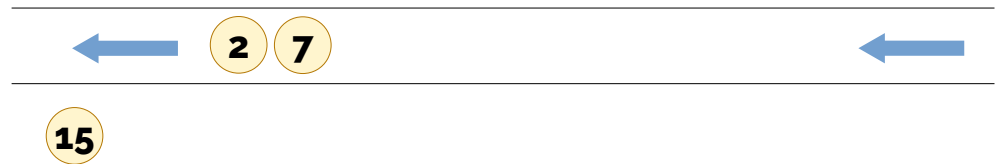


- Insertamos la raíz en la cola.
- Repetimos:
 - Sacamos nodo de la cola.
 - Visitamos ese nodo.
 - Insertamos sus hijos en la cola.

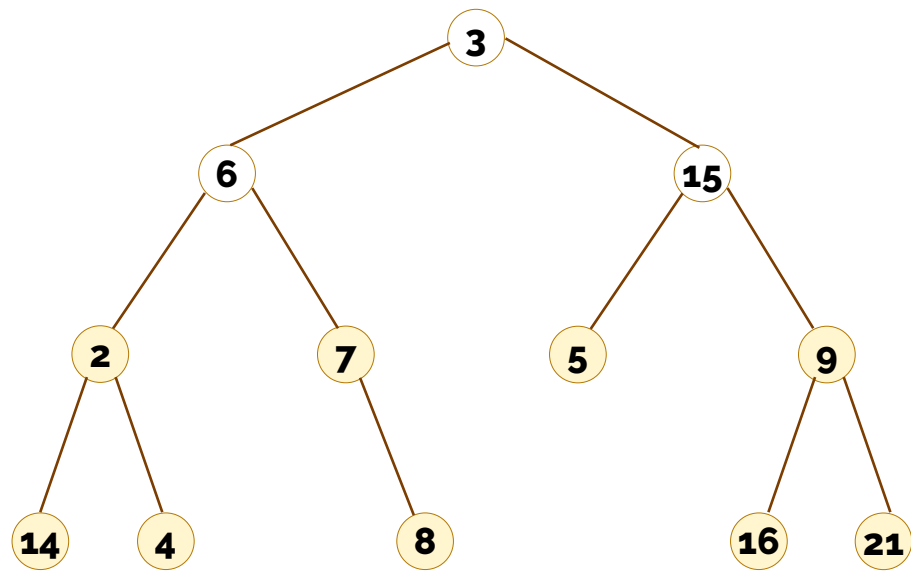
Algoritmo de recorrido en anchura



- Insertamos la raíz en la cola.
- Repetimos:
 - Sacamos nodo de la cola.
 - Visitamos ese nodo.
 - Insertamos sus hijos en la cola.

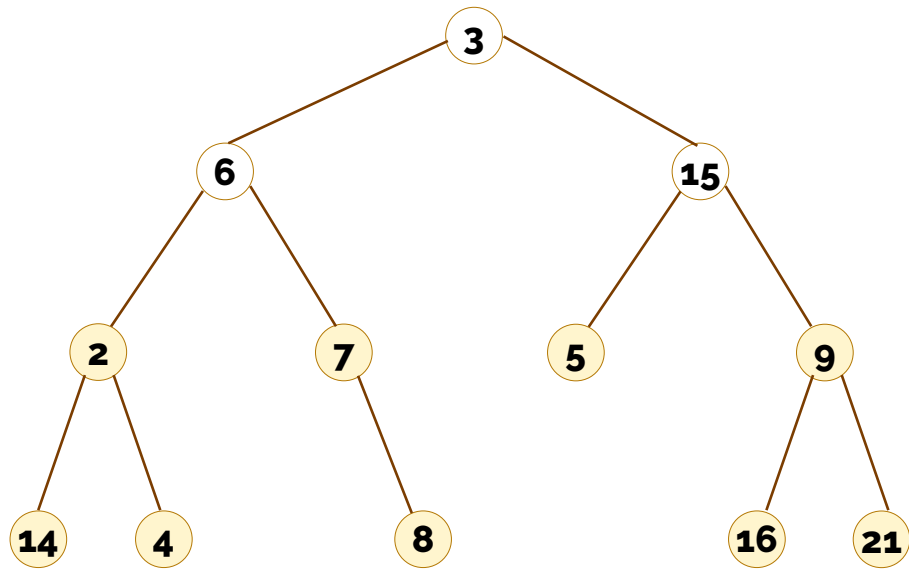


Algoritmo de recorrido en anchura



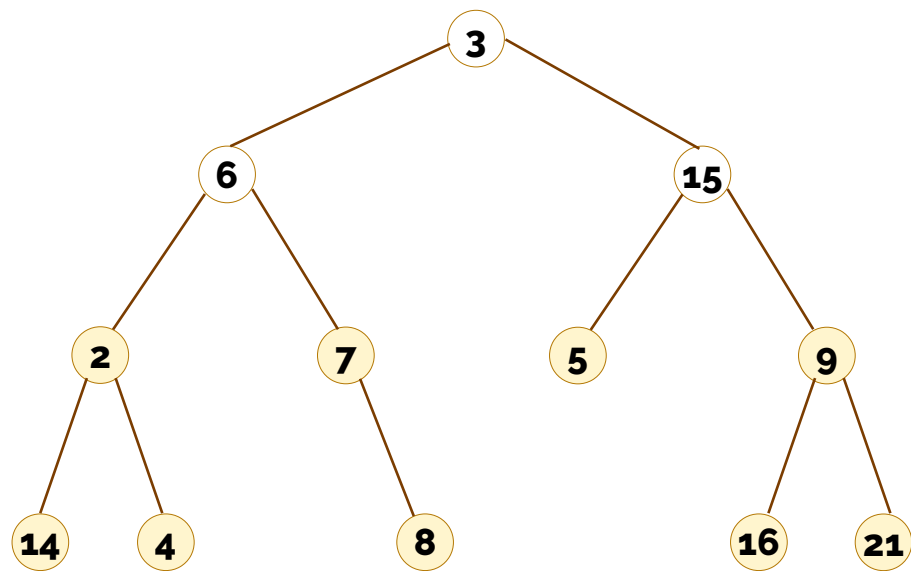
- Insertamos la raíz en la cola.
- Repetimos:
 - Sacamos nodo de la cola.
 - Visitamos ese nodo.
 - Insertamos sus hijos en la cola.

Algoritmo de recorrido en anchura

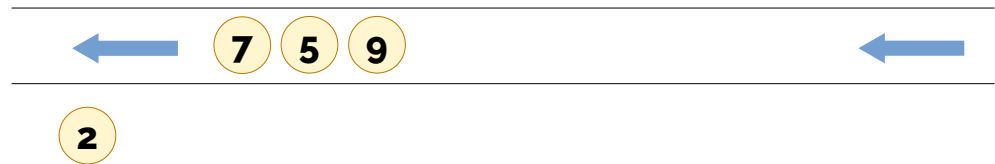


- Insertamos la raíz en la cola.
- Repetimos:
 - Sacamos nodo de la cola.
 - Visitamos ese nodo.
 - Insertamos sus hijos en la cola.

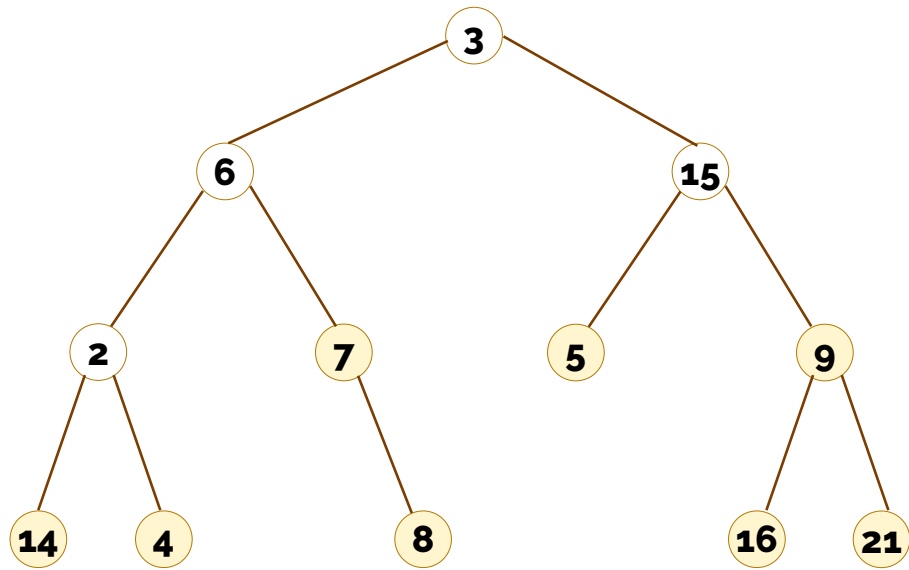
Algoritmo de recorrido en anchura



- Insertamos la raíz en la cola.
- Repetimos:
 - Sacamos nodo de la cola.
 - Visitamos ese nodo.
 - Insertamos sus hijos en la cola.



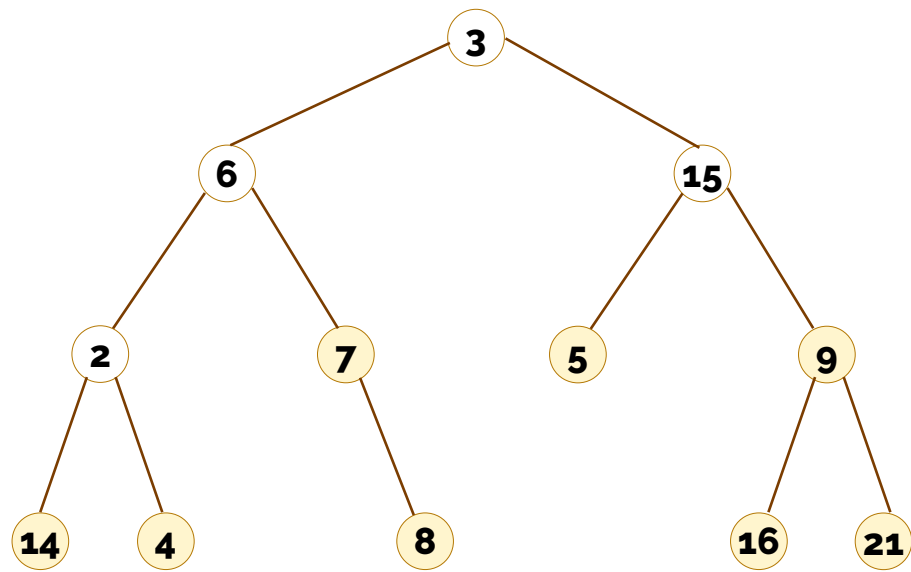
Algoritmo de recorrido en anchura



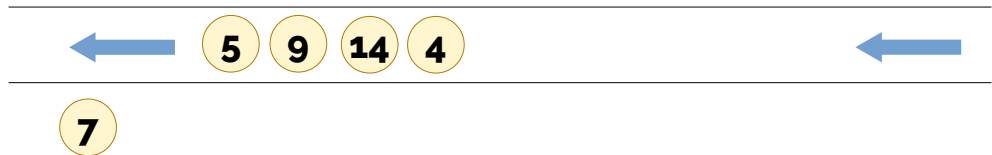
- Insertamos la raíz en la cola.
- Repetimos:
 - Sacamos nodo de la cola.
 - Visitamos ese nodo.
 - Insertamos sus hijos en la cola.



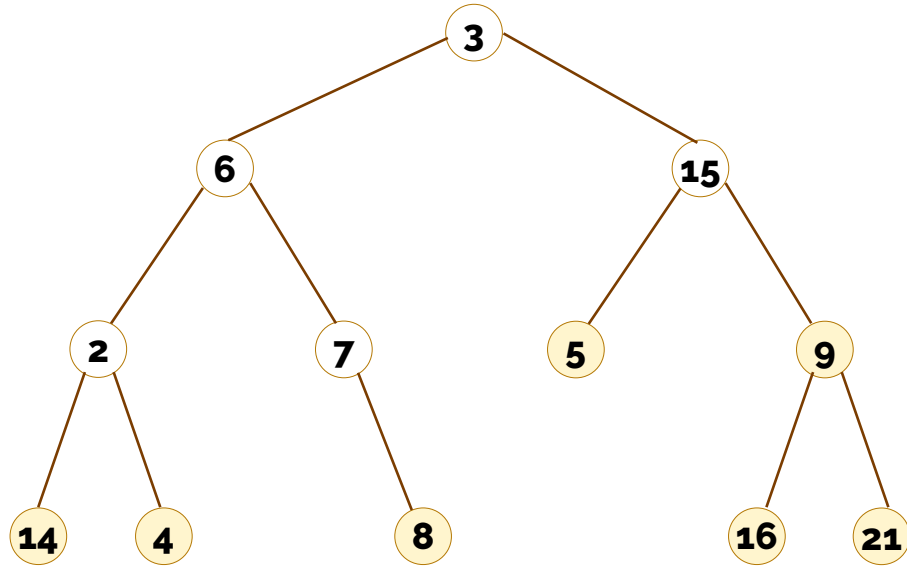
Algoritmo de recorrido en anchura



- Insertamos la raíz en la cola.
- Repetimos:
 - Sacamos nodo de la cola.
 - Visitamos ese nodo.
 - Insertamos sus hijos en la cola.

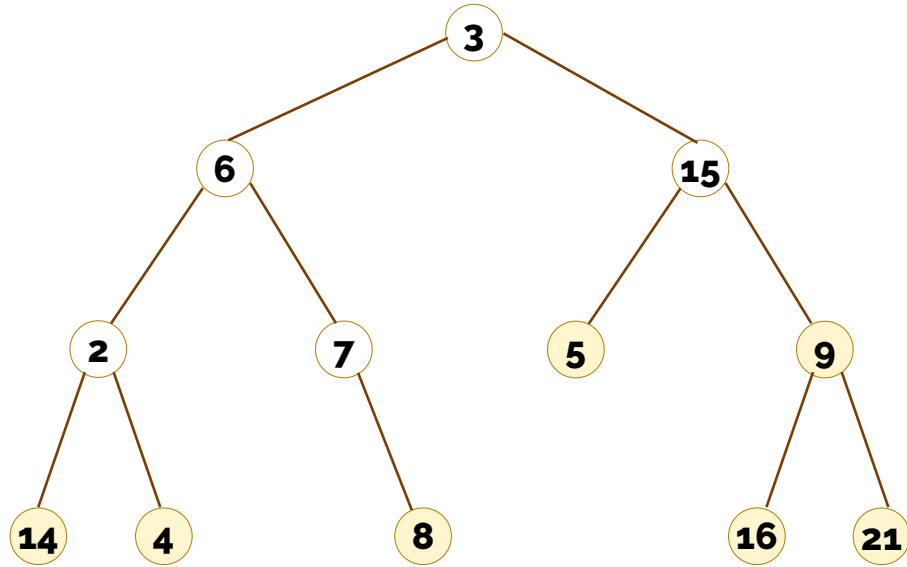


Algoritmo de recorrido en anchura

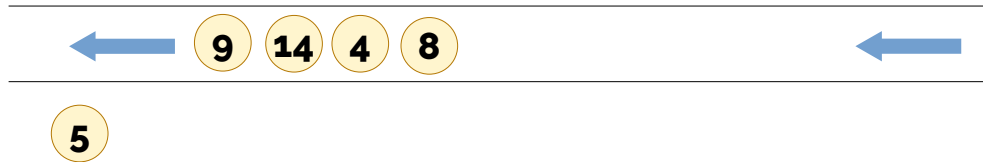


- Insertamos la raíz en la cola.
- Repetimos:
 - Sacamos nodo de la cola.
 - Visitamos ese nodo.
 - Insertamos sus hijos en la cola.

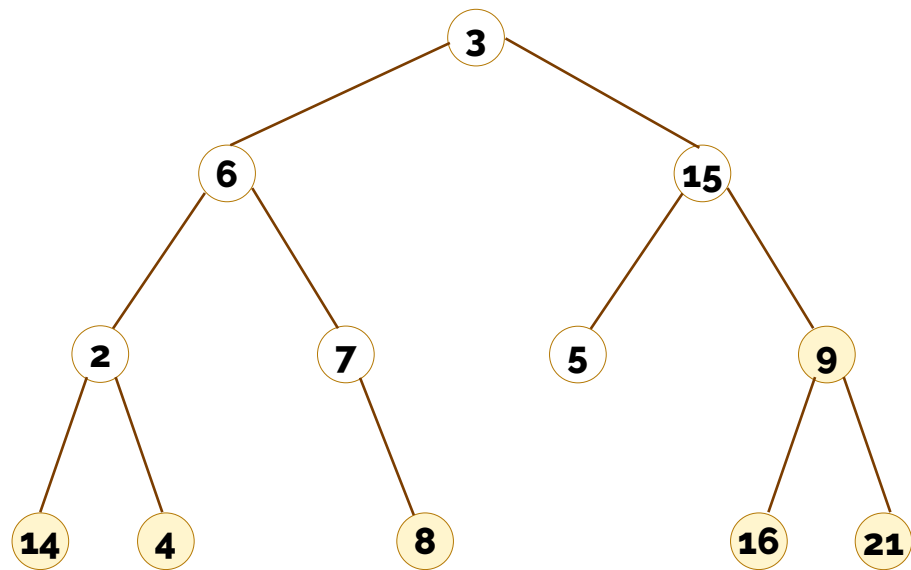
Algoritmo de recorrido en anchura



- Insertamos la raíz en la cola.
- Repetimos:
 - Sacamos nodo de la cola.
 - Visitamos ese nodo.
 - Insertamos sus hijos en la cola.



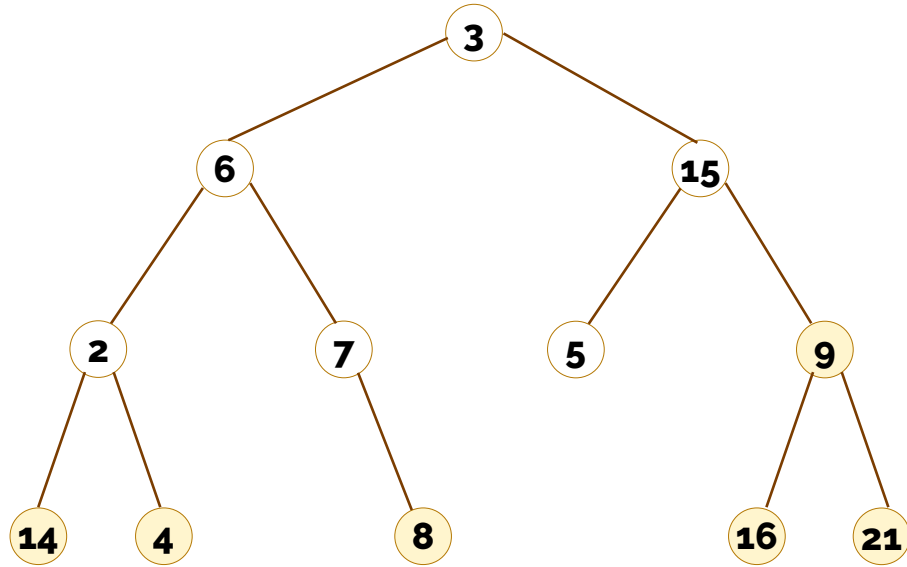
Algoritmo de recorrido en anchura



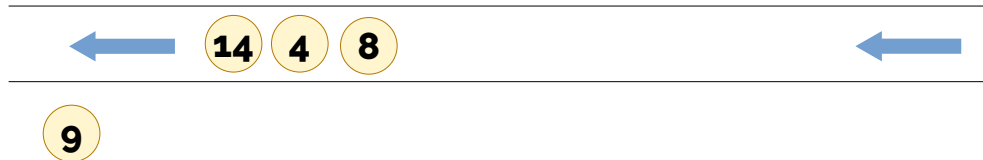
- Insertamos la raíz en la cola.
- Repetimos:
 - Sacamos nodo de la cola.
 - Visitamos ese nodo.
 - Insertamos sus hijos en la cola.



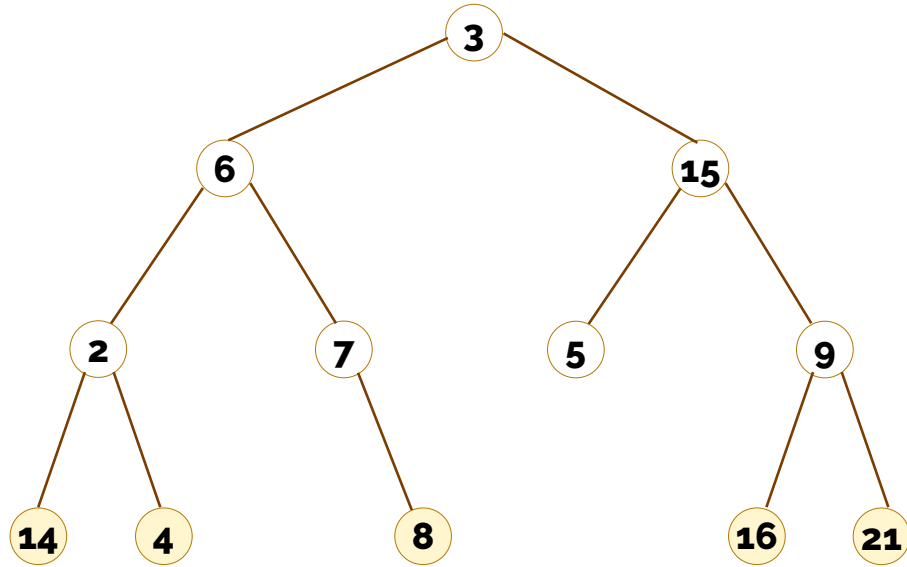
Algoritmo de recorrido en anchura



- Insertamos la raíz en la cola.
- Repetimos:
 - Sacamos nodo de la cola.
 - Visitamos ese nodo.
 - Insertamos sus hijos en la cola.

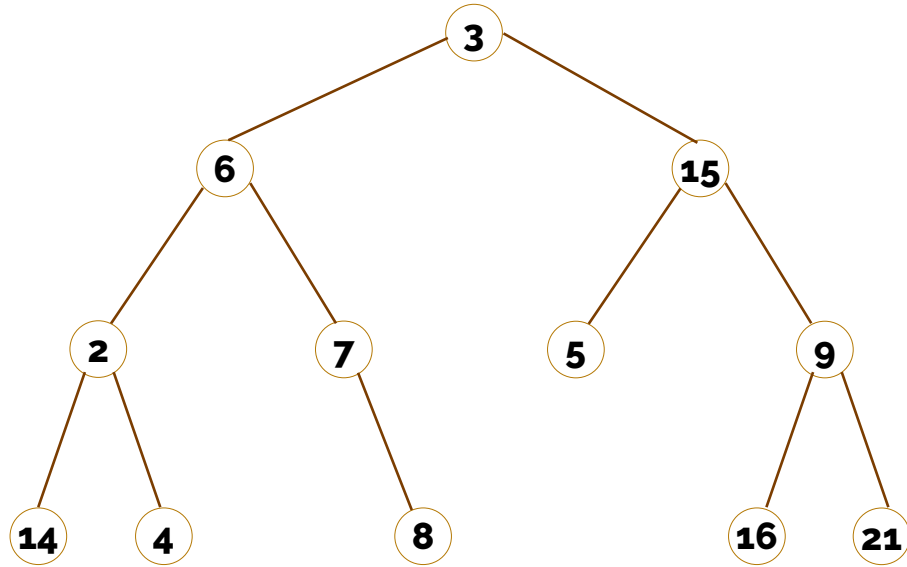


Algoritmo de recorrido en anchura



- Insertamos la raíz en la cola.
- Repetimos:
 - Sacamos nodo de la cola.
 - Visitamos ese nodo.
 - Insertamos sus hijos en la cola.

Algoritmo de recorrido en anchura



- Insertamos la raíz en la cola.
 - Repetimos:
 - Sacamos nodo de la cola.
 - Visitamos ese nodo.
 - Insertamos sus hijos en la cola.
- hasta que la cola esté vacía.

Código de levelorder

```
template<typename T>
void BinTree<T>::levelorder() const {
    std::queue<NodePointer> pending;
    pending.push(root_node);

    while (!pending.empty()) {
        NodePointer current = pending.front();
        pending.pop();
        std::cout << current->elem << " ";
        if (current->left != nullptr) {
            pending.push(current->left);
        }
        if (current->right != nullptr) {
            pending.push(current->right);
        }
    }
}
```

- Insertamos la raíz en la cola.
 - Repetimos:
 - Sacamos nodo de la cola.
 - Visitamos ese nodo.
 - Insertamos sus hijos en la cola.
- hasta que la cola esté vacía.

Código de levelorder

```
template<typename T>
void BinTree<T>::levelorder() const {
    std::queue<NodePointer> pending;
    if (root_node != nullptr) {
        pending.push(root_node);
    }
    while (!pending.empty()) {
        NodePointer current = pending.front();
        pending.pop();
        std::cout << current->elem << " ";
        if (current->left != nullptr) {
            pending.push(current->left);
        }
        if (current->right != nullptr) {
            pending.push(current->right);
        }
    }
}
```

- Añadimos una guarda en el caso en el que el árbol esté vacío.



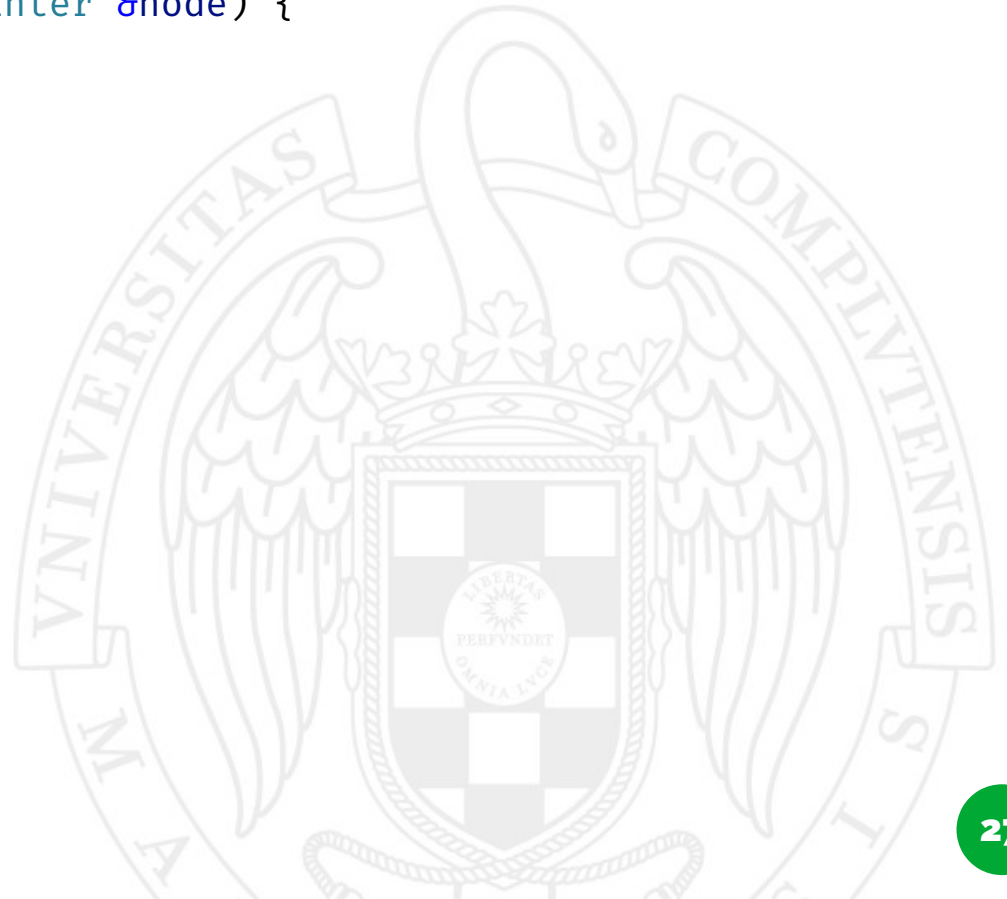
Ejemplo

```
int main() {  
    BinTree<int> tree {{{ 9 }, 4, { 5 }}, 7, {{ }, 4, { 6 }}};  
  
    std::cout << "Recorrido por niveles: " << std::endl;  
    tree.levelorder();  
    std::cout << std::endl;  
  
    return 0;  
}
```

Recorrido por niveles:
7 4 4 9 5 6

Método auxiliar postorder

```
template<typename T>
void BinTree<T>::postorder(const NodePointer &node) {
    if (node != nullptr) {
        postorder(node->left);
        postorder(node->right);
        std::cout << node->elem << " ";
    }
}
```



Ejemplo

```
int main() {  
    BinTree<int> tree = {{{ 9 }, 4, { 5 }}, 7, {{{ 10 }, 4, { 6 }}}};  
  
    std::vector<int> vec;  
    std::cout << "Recorrido en preorden: " << std::endl;  
    tree.preorder();  
    std::cout << std::endl;  
  
    std::cout << "Recorrido en inorden: " << std::endl;  
    tree.inorder();  
    std::cout << std::endl;  
  
    std::cout << "Recorrido en postorden: " << std::endl;  
    tree.postorder();  
    std::cout << std::endl;  
  
    return 0;  
}
```

Recorrido en preorden:
7 4 9 5 4 10 6
Recorrido en inorden:
9 4 5 7 10 4 6
Recorrido en postorden:
9 5 4 10 6 4 7