Boundary traversal in C++

```cpp
#include <iostream>
#include <vector>
using namespace std;

// Definition of the Node class
class Node {
public:
    int key;
    Node* left;
    Node* right;

    Node(int item) {
        key = item;
        left = right = nullptr;
    }
};

// Utility function to check if a node is a leaf node
bool isLeaf(Node* root) {
    return (root->left == nullptr && root->right ==
nullptr);
}

// Function to add nodes of the left boundary
(excluding the leaf node itself)
void addLeftBoundary(Node* root, vector<int>&
res) {
    Node* cur = root->left;
    while (cur != nullptr) {
        if (!isLeaf(cur))
            res.push_back(cur->key);
        if (cur->left != nullptr)
            cur = cur->left;
        else
            cur = cur->right;
    }
}

// Function to add nodes of the right boundary
(excluding the leaf node itself)
void addRightBoundary(Node* root, vector<int>&
res) {
    Node* cur = root->right;
    vector<int> tmp;
    while (cur != nullptr) {
        if (!isLeaf(cur))
            tmp.push_back(cur->key);
        if (cur->right != nullptr)
            cur = cur->right;
        else
            cur = cur->left;
    }
    for (int i = tmp.size() - 1; i >= 0; --i) {
        res.push_back(tmp[i]);
    }
}

// Function to add all leaf nodes in left-to-right
order
```

Binary Tree Structure

Here's the tree again for reference:

```
        1
       / \
      2   7
     /     \
    3       8
     \     /
      4   9
     / \ / \
    5  6 10 11
```

✅ Step-by-Step Tabular Dry Run

1. 🟩 Root Node

Step	Node Visited	Is Leaf?	Action	Vector State
1	1	No	Add to result	[1]

2. 🟨 Left Boundary (excluding leaves)

Traversal path: 2 → 3 → 4 (stop before leaf nodes 5, 6)

Step	Node Visited	Is Leaf?	Action	Vector State
2	2	No	Add to result	[1, 2]
3	3	No	Add to result	[1, 2, 3]
4	4	No	Add to result	[1, 2, 3, 4]

3. 🟦 Leaf Nodes (from left to right)

Leaf nodes: 5, 6, 10, 11

Step	Node Visited	Is Leaf?	Action	Vector State
5	5	Yes	Add to result	[1, 2, 3, 4, 5]
6	6	Yes	Add to result	[1, 2, 3, 4, 5, 6]
7	10	Yes	Add to	[1, 2, 3, 4, 5, 6,

```cpp
void addLeaves(Node* root, vector<int>& res) {
    if (isLeaf(root)) {
        res.push_back(root->key);
        return;
    }
    if (root->left != nullptr)
        addLeaves(root->left, res);
    if (root->right != nullptr)
        addLeaves(root->right, res);
}

// Function to perform boundary traversal and
return the result as vector
vector<int> printBoundary(Node* node) {
    vector<int> ans;
    if (!isLeaf(node))
        ans.push_back(node->key);
    addLeftBoundary(node, ans);
    addLeaves(node, ans);
    addRightBoundary(node, ans);
    return ans;
}

int main() {
    // Constructing the binary tree
    Node* root = new Node(1);
    root->left = new Node(2);
    root->left->left = new Node(3);
    root->left->left->right = new Node(4);
    root->left->left->right->left = new Node(5);
    root->left->left->right->right = new Node(6);
    root->right = new Node(7);
    root->right->right = new Node(8);
    root->right->right->left = new Node(9);
    root->right->right->left->left = new Node(10);
    root->right->right->left->right = new Node(11);

    // Performing boundary traversal
    vector<int> boundaryTraversal =
printBoundary(root);

    // Printing the result
    cout << "The Boundary Traversal is : ";
    for (int i = 0; i < boundaryTraversal.size(); i++)
{
        cout << boundaryTraversal[i] << " ";
    }
    cout << endl;

    return 0;
}
```

Step	Node Visited	Is Leaf?	Action	Vector State
			result	10]
8	11	Yes	Add to result	[1, 2, 3, 4, 5, 6, 10, 11]

4. 🟥 Right Boundary (excluding leaves) — reversed

Traversal path: 7 → 8 → 9 (reverse order, ignore 10 and 11)

Step	Node Visited	Is Leaf?	Action (store in temp, then reverse)	Temporary Stack	Vector State (after reverse append)
9	7	No	Push to temp	[7]	
10	8	No	Push to temp	[7, 8]	
11	9	No	Push to temp	[7, 8, 9]	
12	--	--	Reverse and append to result		[1, 2, 3, 4, 5, 6, 10, 11, 9, 8, 7]

🎯 Final Result
Boundary Traversal: [1, 2, 3, 4, 5, 6, 10, 11, 9, 8, 7]

Boundary Traversal: [1, 2, 3, 4, 5, 6, 10, 11, 9, 8, 7]