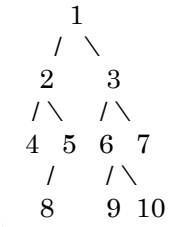
Iterative Inorder in C++

```
#include <iostream>
#include <vector>
#include <stack>
using namespace std;
// TreeNode structure definition
struct TreeNode {
    int key;
    TreeNode* left;
    TreeNode* right;
    TreeNode(int x) {
        key = x;
        left = nullptr;
        right = nullptr;
    }
};

// Function to perform iterative inorder traversal
vector<int> inOrderTrav(TreeNode* root) {
    vector<int> inOrder;
    stack<TreeNode*> s;
    TreeNode* curr = root;

    while (true) {
        if (curr != nullptr) {
            s.push(curr);
            curr = curr->left;
        } else {
            if (s.empty()) break;
            curr = s.top();
            inOrder.push_back(curr->key);
            s.pop();
            curr = curr->right;
        }
    }
    return inOrder;
}

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

    // Perform iterative inorder traversal
    vector<int> inOrder = inOrderTrav(root);

    // Print the result
    cout << "The inorder traversal is : ";
    for (int i = 0; i < inOrder.size(); i++) {
        cout << inOrder[i] << " ";
    }
    cout << endl;
```

Tree Structure:



Dry Run Table

Step	Current Node (curr)	Stack (top → bottom)	Action	Output (inOrder)
1	1		Push 1, move to left	
2	2	1	Push 2, move to left	
3	4	2 → 1	Push 4, move to left	
4	nullptr	4 → 2 → 1	Pop 4, visit	4
5	nullptr (right of 4)	2 → 1	Pop 2, visit	4 2
6	5	1	Push 5, move to left	4 2
7	8	5 → 1	Push 8, move to left	4 2
8	nullptr	8 → 5 → 1	Pop 8, visit	4 2 8
9	nullptr (right of 8)	5 → 1	Pop 5, visit	4 2 8 5
10	nullptr (right of 5)	1	Pop 1, visit	4 2 8 5 1
11	3		Push 3, move to left	4 2 8 5 1
12	6	3	Push 6, move to left	4 2 8 5 1

```
    return 0;
}
```

13	nullptr	6 → 3	Pop 6, visit	4 2 8 5 1 6
14	nullptr (right of 6)	3	Pop 3, visit	4 2 8 5 1 6 3
15	7		Push 7, move to left	4 2 8 5 1 6 3
16	9	7	Push 9, move to left	4 2 8 5 1 6 3
17	nullptr	9 → 7	Pop 9, visit	4 2 8 5 1 6 3 9
18	nullptr (right of 9)	7	Pop 7, visit	4 2 8 5 1 6 3 9 7
19	10		Push 10, move to left	4 2 8 5 1 6 3 9 7
20	nullptr	10	Pop 10, visit	4 2 8 5 1 6 3 9 7 10

✅ **Final Output:**

The inorder traversal is : 4 2 8 5 1 6 3 9 7 10

The inorder traversal is : 4 2 8 5 1 6 3 9 7 10