Level Order in C++

```
#include <iostream>
#include <vector>
#include <queue>
#include <stack>

using namespace std;

// Node class definition
class Node {
public:
    int data;
    vector<Node*> children;

    Node(int val) {
        data = val;
    }
};

// Function to construct the tree from the given array
Node* construct(vector<int>& arr) {
    Node* root = nullptr;
    stack<Node*> st;

    for (int i = 0; i < arr.size(); ++i) {
        if (arr[i] == -1) {
            st.pop();
        } else {
            Node* t = new Node(arr[i]);

            if (!st.empty()) {
                st.top()->children.push_back(t);
            } else {
                root = t;
            }

            st.push(t);
        }
    }

    return root;
}

// Function for level order traversal
void levelOrder(Node* node) {
    if (!node)
        return;

    queue<Node*> q;
    q.push(node);

    while (!q.empty()) {
        Node* f = q.front();
        q.pop();

        cout << f->data << " ";

        for (Node* child : f->children) {
            q.push(child);
        }
    }
```

Input Array:
{24, 10, 20, 50, -1, 60, -1, -1, 30, 70, -1, 80, 110, -1, 120, -1, -1, 90, -1, -1, 40, 100, -1, -1, -1}

🔧 Tree Construction Process (construct() function):

Using a **stack**, we construct the tree as follows:

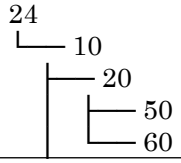
Step	arr[i]	Action	Stack Top (parent)	Node Created	Description
0	24	Create root, push to stack	—	24	Root node
1	10	Create, add to 24, push	24	10	24 → 10
2	20	Create, add to 10, push	10	20	10 → 20
3	50	Create, add to 20, push	20	50	20 → 50
4	-1	Pop 50	20	—	50 done
5	60	Create, add to 20, push	20	60	20 → 60
6	-1	Pop 60	20	—	60 done
7	-1	Pop 20	10	—	20 done
8	30	Create, add to 10, push	10	30	10 → 30
9	70	Create, add to 30, push	30	70	30 → 70
10	-1	Pop 70	30	—	70 done
11	80	Create, add to	30	80	30 → 80

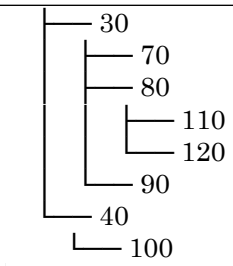
```
    cout << "." << endl;
}

// Main function
int main() {
    vector<int> arr = {24, 10, 20, 50, -1, 60, -1, -1,
30, 70, -1, 80, 110, -1, 120, -1, -1, 90, -1, -1, 40,
100, -1, -1, -1};

    Node* root = construct(arr);
    levelOrder(root);

    return 0;
}
```

		30, push			
12	110	Create, add to 80, push	80	110	80 → 110
13	-1	Pop 110	80	—	110 done
14	120	Create, add to 80, push	80	120	80 → 120
15	-1	Pop 120	80	—	120 done
16	-1	Pop 80	30	—	80 done
17	90	Create, add to 30, push	30	90	30 → 90
18	-1	Pop 90	30	—	90 done
19	-1	Pop 30	10	—	30 done
20	40	Create, add to 10, push	10	40	10 → 40
21	100	Create, add to 40, push	40	100	40 → 100
22	-1	Pop 100	40	—	100 done
23	-1	Pop 40	10	—	40 done
24	-1	Pop 10	24	—	10 done

✅ Final tree root is 24

🌳 **Tree Structure (for Visualization)**





🔄 Level Order Traversal Output

Traverses level-by-level:

Queue Contents	Output
24	24
10	10
20, 30, 40	20
50, 60, 70, 80, 90, 100	30
—	40
—	50
—	60
—	70
110, 120	80
—	90
—	100
—	110
—	120

✅ Final Output:

24 10 20 30 40 50 60 70 80 90 100 110 120 .

24 10 20 30 40 50 60 70 80 90 100 110 120 .