|  |  |
| --- | --- |
| **Diagonal Order in C++** | |
| #include <iostream>  #include <vector>  #include <queue>  using namespace std;  // TreeNode structure definition  struct TreeNode {  int val;  TreeNode\* left;  TreeNode\* right;  TreeNode(int x) {  val = x;  left = nullptr;  right = nullptr;  }  };  // Function to perform diagonal order traversal of a binary tree  vector<vector<int>> diagonalOrder(TreeNode\* root) {  vector<vector<int>> ans;  if (root == nullptr) return ans;  queue<TreeNode\*> que;  que.push(root);  while (!que.empty()) {  int size = que.size();  std::vector<int> smallAns;  while (size--) {  TreeNode\* node = que.front();  que.pop();  while (node != nullptr) {  smallAns.push\_back(node->val);  if (node->left) que.push(node->left);  node = node->right;  }  }  ans.push\_back(smallAns);  }  return ans;  }  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->right->left = new TreeNode(6);  root->right->right = new TreeNode(7);  // Calling diagonalOrder function and printing the result  vector<vector<int>> ans = diagonalOrder(root);  cout << "Diagonal Order Traversal:\n";  for (const auto level : ans) {  for (int num : level) {  cout << num << " ";  }  cout << "\n";  }  // Deallocating memory to avoid memory leaks  delete root->right->right;  delete root->right->left;  delete root->left->right;  delete root->left->left;  delete root->right;  delete root->left;  delete root;  return 0;  } | **Tree Structure:**  1  / \  2 3  / \ / \  4 5 6 7  **🔸 Diagonal View Intuition:**   * Diagonal lines go **from top-right to bottom-left**, i.e., every time you go to .right, you stay on the same diagonal. * Every time you go to .left, you move to the **next diagonal**.   **✅ Dry Run Table:**  We'll simulate the queue and how the diagonal groups are formed.   | **Iteration** | **Queue (Before)** | **Extracted** | **Collected (Diagonal)** | **Queue (After pushing lefts)** | | --- | --- | --- | --- | --- | | 1 | [1] | 1 → 3 → 7 | [1, 3, 7] | [2, 6] | | 2 | [2, 6] | 2 → 5 | [2, 5] | [4] | | 3 | [4] | 4 | [4] | [] |   **🔹 Final Output:**  Diagonal Order Traversal:  1 3 7  2 5  4  **💡 Breakdown:**   * Diagonal 0 → 1 → 3 → 7 * Diagonal 1 → 2 → 5 * Diagonal 2 → 4 |
| Diagonal Order Traversal:  1 3 7  2 5 6  4 | |