|  |  |
| --- | --- |
| **Array 2 BST in C++** | |
| #include <iostream>  #include <queue>  using namespace std;  class Node {  public:  int key;  Node\* left;  Node\* right;  Node(int item) {  key = item;  left = nullptr;  right = nullptr;  }  };  Node\* SortedArrayToBST(int arr[], int start, int end) {  if (start > end) {  return nullptr;  }  int mid = (start + end) / 2;  Node\* root = new Node(arr[mid]);  root->left = SortedArrayToBST(arr, start, mid - 1);  root->right = SortedArrayToBST(arr, mid + 1, end);  return root;  }  void printLevelWise(Node\* root) {  if (root == nullptr) {  return;  }  queue<Node\*> q;  q.push(root);  while (!q.empty()) {  int size = q.size();  for (int i = 0; i < size; i++) {  Node\* current = q.front();  q.pop();  cout << current->key << " ";  if (current->left != nullptr) {  q.push(current->left);  }  if (current->right != nullptr) {  q.push(current->right);  }  }  cout << endl;  }  }  int main() {  int arr[] = {1, 2, 3, 4, 5, 6};  int n = sizeof(arr) / sizeof(arr[0]);  Node\* root = SortedArrayToBST(arr, 0, n - 1);  cout << "Level order traversal of constructed BST:" << endl;  printLevelWise(root);  return 0;  } | **Input Array:**  arr = {1, 2, 3, 4, 5, 6}  **🧠 Algorithm: SortedArrayToBST**  The function picks the **middle element** as the root recursively:   * Left subtree from elements left of mid * Right subtree from elements right of mid   **🌳 Constructed BST:**  Here's the tree built step-by-step:  Index: 0 1 2 3 4 5  Array: 1 2 3 4 5 6  Step-by-step recursive mid values:  - Root: mid = (0+5)/2 = 2 → Node(3)  - Left child: mid = (0+1)/2 = 0 → Node(1)  - Right of 1: mid = (1+1)/2 = 1 → Node(2)  - Right child: mid = (3+5)/2 = 4 → Node(5)  - Left of 5: mid = (3+3)/2 = 3 → Node(4)  - Right of 5: mid = (5+5)/2 = 5 → Node(6)  Final BST:  3  / \  1 5  \ / \  2 4 6  **🔁 Dry Run of printLevelWise**   | **Level** | **Queue Contents** | **Printed Nodes** | | --- | --- | --- | | 1 | [3] | 3 | | 2 | [1, 5] | 1 5 | | 3 | [2, 4, 6] | 2 4 6 |   **✅ Final Output:**  Level order traversal of constructed BST:  3  1 5  2 4 6 |
| Level order traversal of constructed BST:  3  1 5  2 4 6 | |