|  |  |
| --- | --- |
| **Deletion in BST in C++** | |
| #include <iostream>  using namespace std;  class Node {  public:  int key;  Node \*left, \*right;  Node(int item) {  key = item;  left = right = nullptr;  }  };  class BST {  public:  Node\* root;  BST() {  root = nullptr;  }  Node\* insert(Node\* root, int x) {  if (root == nullptr) {  return new Node(x);  }  if (x < root->key) {  root->left = insert(root->left, x);  } else if (x > root->key) {  root->right = insert(root->right, x);  }  return root;  }  void inorder(Node\* root) {  if (root != nullptr) {  inorder(root->left);  cout << root->key << " ";  inorder(root->right);  }  }  Node\* deleteNode(Node\* root, int x) {  if (root == nullptr) {  return root;  }  if (x < root->key) {  root->left = deleteNode(root->left, x);  } else if (x > root->key) {  root->right = deleteNode(root->right, x);  } else {  if (root->left == nullptr) {  Node\* temp = root->right;  delete root;  return temp;  } else if (root->right == nullptr) {  Node\* temp = root->left;  delete root;  return temp;  }  Node\* succ = getSuccessor(root->right);  root->key = succ->key;  root->right = deleteNode(root->right, succ->key);  }  return root;  }  Node\* getSuccessor(Node\* root) {  Node\* curr = root;  while (curr != nullptr && curr->left != nullptr) {  curr = curr->left;  }  return curr;  }  };  int main() {  BST tree;  tree.root = tree.insert(tree.root, 10);  tree.insert(tree.root, 30);  tree.insert(tree.root, 20);  tree.insert(tree.root, 40);  tree.insert(tree.root, 70);  tree.insert(tree.root, 60);  tree.insert(tree.root, 80);  cout << "Inorder traversal before deletion: ";  tree.inorder(tree.root);  cout << endl;  tree.deleteNode(tree.root, 20);  cout << "Inorder traversal after deletion: ";  tree.inorder(tree.root);  cout << endl;  return 0;  } | **Initial Tree Structure** You inserted values in this order:  10, 30, 20, 40, 70, 60, 80  Resulting BST:  10  \  30  / \  20 40  \  70  / \  60 80 🔁 **Dry Run of** deleteNode(root, 20)  | **Step** | **Function Call** | **Current Node** | **Comparison** | **Action Taken** | | --- | --- | --- | --- | --- | | 1 | deleteNode(root, 20) | 10 | 20 > 10 | Go right → call deleteNode(30, 20) | | 2 | deleteNode(30, 20) | 30 | 20 < 30 | Go left → call deleteNode(20, 20) | | 3 | deleteNode(20, 20) | 20 | Match found | Node with no children, return nullptr | | 4 | Return to Step 2 | 30 | Set left = nullptr | 20 is deleted from left of 30 | | 5 | Return to Step 1 | 10 | Set right = result | Subtree rooted at 30 is updated after deletion |  ✅ **Final Tree Structure (After Deletion)** 10  \  30  \  40  \  70  / \  60 80 📤 **Inorder Traversals**  | **State** | **Inorder Output** | | --- | --- | | Before Deletion | 10 20 30 40 60 70 80 | | After Deletion | 10 30 40 60 70 80 | |
| Inorder traversal before deletion: 10 20 30 40 60 70 80  Inorder traversal after deletion: 10 30 40 60 70 80 | |