|  |  |
| --- | --- |
| **Size,Sum,Max,Min,Height in C++** | |
| #include <iostream>  #include <algorithm>  #include <climits>// for std::max  using namespace std;  // Definition of a binary tree node  struct Node {  int data;  Node\* left;  Node\* right;  Node(int data, Node\* left = nullptr, Node\* right = nullptr) {  this->data = data;  this->left = left;  this->right = right;  }  };  // Function to calculate the size (number of nodes) of the binary tree  int size(Node\* node) {  if (node == nullptr) {  return 0;  } else {  return 1 + size(node->left) + size(node->right);  }  }  // Function to calculate the sum of all nodes in the binary tree  int sum(Node\* node) {  if (node == nullptr) {  return 0;  } else {  int lsum = sum(node->left);  int rsum = sum(node->right);  return node->data + lsum + rsum;  }  }  // Function to find the maximum value in the binary tree  int max(Node\* node) {  if (node == nullptr) {  return INT\_MIN; // from <climits> for INT\_MIN  } else {  int lmax = max(node->left);  int rmax = max(node->right);  return std::max(node->data, std::max(lmax, rmax));  }  }  // Function to calculate the height of the binary tree  int height(Node\* node) {  if (node == nullptr) {  return -1;  } else {  int lh = height(node->left);  int rh = height(node->right);  return 1 + std::max(lh, rh);  }  }  // Function to display the binary tree (inorder traversal)  void display(Node\* node) {  if (node == nullptr) {  return;  }  display(node->left);  cout << node->data << " ";  display(node->right);  }  int main() {  // Hardcoded tree construction  Node\* root = new Node(50);  root->left = new Node(25);  root->left->left = new Node(12);  root->left->right = new Node(37);  root->left->right->left = new Node(30);  root->right = new Node(75);  root->right->left = new Node(62);  root->right->left->right = new Node(70);  root->right->right = new Node(87);  // Calculating size, sum, max value, and height  int treeSize = size(root);  int treeSum = sum(root);  int treeMax = max(root);  int treeHeight = height(root);  // Displaying results  cout << "Size of the binary tree: " << treeSize << endl;  cout << "Sum of all nodes in the binary tree: " << treeSum << endl;  cout << "Maximum value in the binary tree: " << treeMax << endl;  cout << "Height of the binary tree: " << treeHeight << endl;  // Displaying the binary tree (inorder traversal)  cout << "Inorder traversal of the binary tree:" << endl;  display(root);  cout << endl;  // Clean up dynamically allocated memory  delete root->right->left->right;  delete root->right->left;  delete root->right;  delete root->left->right->left;  delete root->left->right;  delete root->left->left;  delete root->left;  delete root;  return 0;  } | **Binary Tree Structure:**  50  / \  25 75  / \ / \  12 37 62 87  / \  30 70  **✅ Expected Outputs:**   | **Function** | **Description** | **Output** | | --- | --- | --- | | size | Number of nodes | 9 | | sum | Sum of all node values | 448 | | max | Maximum value in the tree | 87 | | height | Height of the tree (edges, not nodes) | 3 | | display | Inorder traversal (left → root → right) | 12 25 30 37 50 62 70 75 87 |   **🧪 Let's go through function results step-by-step:**  **1. size(root):**   * Total nodes = 9   **2. sum(root):**  = 50 + sum(25 subtree) + sum(75 subtree)  = 50 + (25 + 12 + 37 + 30) + (75 + 62 + 70 + 87)  = 50 + 104 + 294  = 448  **3. max(root):**   * Max in left subtree = max(25, 12, 37, 30) = 37 * Max in right subtree = max(75, 62, 70, 87) = 87 * Final max = max(50, 37, 87) = **87**   **4. height(root):**   * Longest path (e.g., 50 → 75 → 62 → 70) has 3 edges → height = **3**   **5. display(root) (Inorder):**  Left subtree (25): 12 25 30 37  Root: 50  Right subtree (75): 62 70 75 87  => Full: 12 25 30 37 50 62 70 75 87  **🖨️ Final Output (Console):**  Size of the binary tree: 9  Sum of all nodes in the binary tree: 448  Maximum value in the binary tree: 87  Height of the binary tree: 3  Inorder traversal of the binary tree:  12 25 30 37 50 62 70 75 87 |
| size of the binary tree: 9  Sum of all nodes in the binary tree: 448  Maximum value in the binary tree: 87  Height of the binary tree: 3  Inorder traversal of the binary tree:  12 25 30 37 50 62 70 75 87 | |