|  |  |
| --- | --- |
| **Tilt in C++** | |
| #include <iostream>  #include <cstdlib> // for abs function  using namespace std;  // Definition of a binary tree node  struct Node {  int data;  Node\* left;  Node\* right;  Node(int item) {  data = item;  left = nullptr;  right = nullptr;  }  };  // Function to display the binary tree (for debugging purposes)  void display(Node\* node) {  if (node == nullptr) {  return;  }  string str = "";  str += (node->left == nullptr) ? "." : to\_string(node->left->data);  str += " <- " + to\_string(node->data) + " -> ";  str += (node->right == nullptr) ? "." : to\_string(node->right->data);  cout << str << endl;  display(node->left);  display(node->right);  }  // Function to calculate the height of the binary tree  int height(Node\* node) {  if (node == nullptr) {  return -1;  }  int lh = height(node->left);  int rh = height(node->right);  return max(lh, rh) + 1;  }  // Global variable to store the tilt of the entire tree  int tilt = 0;  // Function to calculate the tilt of the binary tree  int calculateTilt(Node\* node) {  if (node == nullptr) {  return 0;  }  int ls = calculateTilt(node->left);  int rs = calculateTilt(node->right);  int ltilt = abs(ls - rs);  tilt += ltilt;  int sum = ls + rs + node->data;  return sum;  }  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);  // Calculate the tilt of the tree  calculateTilt(root);  // Output the tilt value  cout << "Tilt of the binary tree: " << tilt << endl;  // Clean up dynamically allocated memory  delete root->left->left;  delete root->left->right->left;  delete root->left->right;  delete root->left;  delete root->right->left->right;  delete root->right->left;  delete root->right->right;  delete root->right;  delete root;  return 0;  } | Tree Structure: 50  / \  25 75  / \ / \  12 37 62 87  / \  30 70 🧮 Dry Run with Tilt Values Let’s go **bottom-up** and calculate each node’s tilt with its left and right subtree sums:   | **Node** | **Left Sum** | **Right Sum** | **Node Tilt = abs(L - R)** | | --- | --- | --- | --- | | 12 | 0 | 0 | 0 | | 30 | 0 | 0 | 0 | | 37 | 30 | 0 | 30 | | 25 | 12 | 67 (37+30) | 55 | | 70 | 0 | 0 | 0 | | 62 | 0 | 70 | 70 | | 87 | 0 | 0 | 0 | | 75 | 132 | 87 | 45 | | 50 | 104 | 294 | 190 |  🔢 Total Tilt: 0 (12)  + 0 (30)  + 30 (37)  + 55 (25)  + 0 (70)  + 70 (62)  + 0 (87)  + 45 (75)  + 190 (50)  = \*\*390\*\* ✅ Output: Tilt of the binary tree: 390 |
| Tilt of the binary tree: 390 | |