《数据结构》实验报告

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 姓名 | 李爽 | 班级 | 22级软件工程专升本1班 | 学号 | 2206831522 |
| 实验名称 | **实验五 树和二叉树** | | | | |
| 实验时间 | 2023.5.4 | | 成绩 | |  |
| 1. **实验目的：**   1.掌握二叉树的逻辑结构；  2.掌握二叉树的建立方法；  3.掌握二叉树遍历算法的应用。  **二、实验工具：**  软件平台：Windows7或以上版本，Visual Studio 2019  **三、实验原理：**  1、算法描述：  （1）建立二叉树的函数：建立二叉树的函数是通过递归来实现的。在递归的过程中，首先会创建当前节点，然后分别递归创建当前节点的左子树和右子树，直到遇到数组中的-1，表示当前节点为叶子节点。最后，该函数返回根节点的指针。  （2）先序遍历函数：先序遍历函数是通过递归来实现的。在递归的过程中，首先会输出当前节点的值，然后递归访问当前节点的左子树和右子树。  （3）中序遍历函数：中序遍历函数也是通过递归来实现的。在递归的过程中，首先会递归访问当前节点的左子树，然后输出当前节点的值，最后递归访问当前节点的右子树。  （4）后序遍历函数：后序遍历函数同样是通过递归来实现的。在递归的过程中，首先会递归访问当前节点的左子树和右子树，然后输出当前节点的值。  **四、实验步骤和内容：**  1. 建立一棵二叉树，树的形态自定，完成以下操作：  （1）输出该二叉树的前序遍历序列；  （2）输出该二叉树的中序遍历序列；  （3）输出该二叉树的后序遍历序列。  #include <iostream>  using namespace std;  // 定义二叉树节点  struct TreeNode {  // 节点值  int val;  // 左子节点  TreeNode\* left;  // 右子节点  TreeNode\* right;  // 构造函数  TreeNode(int x) : val(x), left(NULL), right(NULL) {}  };  // 建立二叉树  TreeNode\* buildTree() {  // 创建根节点，值为1  TreeNode\* root = new TreeNode(1);  // 创建左右子节点，值分别为2和3  root->left = new TreeNode(2);  root->right = new TreeNode(3);  // 创建左子节点的左右子节点，值分别为4和5  root->left->left = new TreeNode(4);  root->left->right = new TreeNode(5);  // 创建右子节点的左右子节点，值分别为6和7  root->right->left = new TreeNode(6);  root->right->right = new TreeNode(7);  // 返回根节点指针  return root;  }  // 前序遍历  void preOrder(TreeNode\* root) {  // 如果节点为空，直接返回  if (root == NULL) return;  // 输出节点值  cout << root->val << " ";  // 遍历左子树  preOrder(root->left);  // 遍历右子树  preOrder(root->right);  }  // 中序遍历  void inOrder(TreeNode\* root) {  // 如果节点为空，直接返回  if (root == NULL) return;  // 遍历左子树  inOrder(root->left);  // 输出节点值  cout << root->val << " ";  // 遍历右子树  inOrder(root->right);  }  // 后序遍历  void postOrder(TreeNode\* root) {  // 如果节点为空，直接返回  if (root == NULL) return;  // 遍历左子树  postOrder(root->left);  // 遍历右子树  postOrder(root->right);  // 输出节点值  cout << root->val << " ";  }  int main() {  // 建立二叉树  TreeNode\* root = buildTree();  // 前序遍历  cout << "前序遍历：";  preOrder(root);  cout << endl;  // 中序遍历  cout << "中序遍历：";  inOrder(root);  cout << endl;  // 后序遍历  cout << "后序遍历：";  postOrder(root);  cout << endl;  return 0;  }  构建的树，如图      **五、实验总结：**  这个实验旨在使用C++语言编写一个二叉树的数据结构，并且实现前序遍历、中序遍历和后序遍历算法。在实验中，我首先定义了一个节点结构，用于存储每个节点的值以及指向左右子节点的指针。接着，我使用C++的类来定义整个二叉树数据结构。  在二叉树建立完成后，我实现了前序遍历、中序遍历和后序遍历算法。前序遍历算法按照根节点、左子树、右子树的顺序遍历二叉树，中序遍历算法按照左子树、根节点、右子树的顺序遍历二叉树，后序遍历算法按照左子树、右子树、根节点的顺序遍历二叉树。  通过这个实验，我深入理解了二叉树的数据结构以及前序遍历、中序遍历和后序遍历算法的实现方法。这些算法对于二叉树的遍历和搜索十分重要，对于日后学习和实践具有重要意义。在C++语言的学习中，我也掌握了类的定义和使用方法，提高了自己的编程技能和能力。  **六、教师评语：** | | | | | |