实验三

|  |  |  |
| --- | --- | --- |
| 姓名： | | 学号： |
| 班级： | | 专业：人工智能 |
| 实验名称： | 二叉树的基本操作 | |
| 实验目的： | 1.掌握二叉树的定义；  2．掌握二叉树的基本操作，如二叉树的建立、遍历、结点个数统计、树的深度计算等。 | |
| 实验要求： | | |
| (1) 程序要添加适当的注释，程序的书写要采用缩进格式。  (2) 程序要具在一定的健壮性，即当输入数据非法时，程序也能适当地做出反应，如插入删除时指定的位置不对等等。  (3) 程序要做到界面友好，在程序运行时用户可以根据相应的提示信息进行操作。  （4）每项任务给出代码及运行结果截图 | | |
| 实验内容： | | |
| 用递归的方法实现以下算法：  1．以二叉链表表示二叉树，建立一棵二叉树（算法5.3）；  2．输出二叉树的中序遍历结果（算法5.1）；  3．输出二叉树的前序遍历结果  4．输出二叉树的后序遍历结果  5．计算二叉树的深度（算法5.5）；  6．统计二叉树的结点个数（算法5.6）；  7．统计二叉树的叶结点个数； | | |
| 实验代码及运行结果截图 | | |
| #include<iostream>  #include<queue>  #define MAXSIZE 100  typedef int TElemtype;  using namespace std;  struct BiTNode {  TElemtype data;  BiTNode\* lchild;  BiTNode\* rchild;  };  typedef BiTNode\* BiTree; //将指向结构体的指针单独命名  BiTNode\* Initialize\_Root(TElemtype data) { //首先需要创建树的根节点（基于结构体的函数）  BiTNode\* root = new BiTNode; //建立新节点  root->data = data; //在根节点的基础上插入根数据和两个分支  root->lchild = nullptr;  root->rchild = nullptr;  return root; //建立完成  }  void insertNode(BiTree& root, TElemtype data) { //操作函数（因此使用void）  if (root == nullptr) { //如果没有根节点就建立一个  root = Initialize\_Root(data);  }  else { //否则就基于节点大小分配左右叶子  if (data < root->data) {  insertNode(root->lchild, data);  }  else {  insertNode(root->rchild, data);  }  }  }  void In\_OrderTraverse(BiTree T) {//这里也是操作，引入了指向结构体的指针T  if (T) { //如果指针T为非空，则中序遍历  In\_OrderTraverse(T->lchild);  cout << T->data;  In\_OrderTraverse(T->rchild);  }  }  void Pre\_OrderTraverse(BiTree T) {  if (T) {  std::cout << T->data;  Pre\_OrderTraverse(T->lchild);  Pre\_OrderTraverse(T->rchild);  }  }  void Post\_OrderTraverse(BiTree T) {  if (T) {  Post\_OrderTraverse(T->lchild);  Post\_OrderTraverse(T->rchild);  std::cout << T->data;  }  }  int depth(BiTree T) {  if (T == NULL) {  return 0;  }else{  int m = depth(T->lchild);  int n = depth(T->rchild);  return max(m, n) + 1; //这里用了一个小技巧避免写if-else,比较出较大的，结果+1  }  }  int Nodecount(BiTree T) {  return (T == nullptr) ? 0 : Nodecount(T->lchild) + Nodecount(T->rchild) + 1;//回顾一下三元表达式  }  //下面是遍历叶节点的代码：叶节点是指没有子节点的节点  int leafCount(BiTree T) {  if (T == nullptr) {  return 0;  }  if (T->lchild == nullptr && T->rchild == nullptr) {  return 1; // 当前节点为叶节点，返回1  }  // 递归统计左右子树的叶节点个数  return leafCount(T->lchild) + leafCount(T->rchild);  }  int main() {  BiTree root = nullptr;  cout << "===========202318140413====周兴==============" << endl;  cout << "============================================="<< endl;  cout << "请输入数字（以字母结束）：" << endl;  TElemtype input;  while (cin >> input) {  if (isalpha(input)) // 判断输入是否为字母  break;  insertNode(root, input); // 插入数字到树中  }  cout << "中序遍历结果：" << endl;  In\_OrderTraverse(root); // 中序遍历并打印树  cout << endl << endl;  cout << "前序遍历结果：" << endl;  Pre\_OrderTraverse(root); // 前序遍历并打印树  cout << endl << endl;  cout << "后序遍历结果：" << endl;  Post\_OrderTraverse(root); // 后序遍历并打印树  cout << endl << endl;  cout << "树的深度：" << depth(root) << endl; // 输出树的深度  cout << "树的节点数：" << Nodecount(root) << endl; // 输出树的节点数  cout << "叶节点个数：" << leafCount(root) << endl; // 输出叶节点个数  cout << endl;  return 0;  } | | |
| 实验结论： | | |
| 知道了链表二叉树的构造方式，对节点和叶节点的统计有了进一步的理解，对三种遍历的过程进行了理解 | | |