山东大学 计算机科学与技术 学院

数据结构与算法 课程实验报告

|  |  |  |  |
| --- | --- | --- | --- |
| 学号：201900161140 | 姓名： 张文浩 | | 班级： 19级人工智能 |
| 实验题目：二叉树操作 | | | |
| 实验学时：2 | | 实验日期： 10.3 | |
| 实验目的：  掌握二叉树的基本概念，链表描述方法；二叉树操作的实现。 | | | |
| 软件开发工具：  Vscode | | | |
| 1. 实验内容   **题目描述**：  接收二叉树前序序列和中序序列(各元素各不相同)，输出该二叉树的后序序列。  **输入输出格式**：  **输入**：  第一行为数字n；  第二行有n个数字，表示二叉树的前序遍历；  第三行有n个数字，表示二叉树的中序遍历。  **输出**：  输出一行，表示该二叉树的后序遍历序列。   1. 数据结构与算法描述 （整体思路描述，所需要的数据结构与算法）   根据中序序列的特性，元素左边的元素属于该节点的左孩子序列，右边则是右孩子序列。所以在递归中，输入的前序遍历左边的节点一定是根节点，所以要找到中序遍历中的改节点，这个节点的左边的节点就是这个根节点的左子树，这个节点的右边的节点就是这个根节点的右子树。然后就可以递归先输出左子树的节点，再输出右子树的节点，再输出这个根节点，就完成了后序输出的过程。   1. 分析与探讨（结果分析，若存在问题，探讨解决问题的途径）   通过本次实验，我实现了二叉树的定义以及包括前序遍历、中序遍历、后序遍历、层次遍历等方法函数。在定义linkedBinaryTree的时候用到了static定义静态函数，了解了静态函数的性质和使用方式，static函数与普通函数作用域不同,仅在本文件。只在当前源文件中使用的函数应该说明为内部函数(static修饰的函数)，内部函数应该在当前源文件中说明和定义。对于可在当前源文件以外使用的函数，应该在一个头文件中说明，要使用这些函数的源文件要包含这个头文件.  static函数在内存中只有一份，普通函数在每个被调用中维持一份拷贝。  在建立二叉树的时候，将整个树的创建拆分为节点之间的连接会使得问题的解决更加地简单。而在其他操作，如前序遍历、中序遍历、后序遍历、统计子树节点数目、计算子树高度中，使用递归的思想解决问题可以使得代码得到很大的简化。而在题目要求2的输出后序遍历中，要注意到输出元素的顺序：左节点，右节点，根节点，并在函数内部安排好子树的划分。  编写题目1的代码时，我起初是将每个函数分开，实现各自的功能的，但在之后发现重复的部分过多，影响了代码的简介性，于是之后采用了重载函数的方式，通过传入某个特定函数的指针，利用同一段代码为遍历赋予不同的功能。在类中定义函数指针之后，类外也需要再定义一次，否则会出现编译失败的问题。函数指针的写法略微有些复杂，尤其是涉及到在其他函数中的调用时，在网络上查询到写法之后我花费了一段时间才将它看懂。编写题目2的代码时，我最开始是先根据前序、中序遍历的顺序建立好二叉树，再进行后序遍历获得结果的，但之后在csdn上看到了直接输出后序排序序列的方法，也就是直接利用递归获取答案。因此我认为学习过程中信息的查询是非常重要的，可以让我们学到更多有用的知识。   1. 附录：实现源代码（本实验的全部源程序代码，程序风格清晰易理解，有充分的注释）   **2.**  #include <iostream>  using namespace std;  int preorder[10000], inorder[10000];  void postorder(int pre, int post, int n)  {      if (n == 1)      {          cout << preorder[pre] << " ";          return;      }      if (n == 0)          return;      int i = 0;      for (; preorder[pre] != inorder[post + i]; i++)          ;      postorder(pre + 1, post, i);//递归输出左子树节点      postorder(pre + i + 1, post + i + 1, n - i - 1);//递归输出右子树节点      cout << preorder[pre] << " ";//输出当前根节点  }  int main()  {      int n;      cin >> n;      for (int i = 0; i < n; i++)      {          cin >> preorder[i];      }      for (int i = 0; i < n; i++)      {          cin >> inorder[i];      }      postorder(0, 0, n);      system("pause");      return 0;  } | | | |