**《数据结构综合设计》实验报告**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 学院 | 人工智能与大数据学院 | 专业 | 虚拟现实技术 | 班级 | 21级2班 | 学生姓名 | 李云豪 |
| 实验  周次 | 5-6 | 实验  日期 | 2023.4.7 | 学时 | 4 | 教师姓名 | 李昊康 |
| 项目名称 | | 树与二叉树的实现及应用 | | | | | |
| 实验  类别 | 🗹验证型实验 🞎设计型实验 🞎综合型实验 🞎其它 | | | | | 成绩：90 | |
| 1. 实验目的及具体要求   实验目的：   1. 掌握二叉树的特点及其存储方式。 2. 掌握二叉树的创建和显示方法。 3. 复习二叉树遍历的概念，掌握二叉树遍历的基本方法。 4. 掌握求二叉树的叶子节点数、树的总结点数和树的深度等基本算法。   具体要求：  用先序法建立一颗二叉树，并能按照广义表表示法显示二叉树结构。  编写先序遍历、中序遍历、后序遍历程序。  编写求二叉树结点数、树的总结点树和深度的程序。  设计选择式菜单，以选择菜单的方式进行操作。   1. 实验仪器、设备和材料   装有并能运行VS2019的电脑。   1. 实验内容、步骤及实验数据记录   #include <stdio.h>  #include <stdlib.h>  // 定义二叉树结点结构体  typedef struct TreeNode {  char data; // 结点数据  struct TreeNode\* left; // 左子树指针  struct TreeNode\* right; // 右子树指针  } TreeNode, \* Tree;  // 先序遍历建立二叉树  void createTree(Tree\* t) {  char ch;  scanf\_s("%c", &ch);  if (ch == '#') {  \*t = NULL;  }  else {  \*t = (Tree)malloc(sizeof(TreeNode));  (\*t)->data = ch;  createTree(&((\*t)->left));  createTree(&((\*t)->right));  }  }  // 广义表表示法输出二叉树  void printTree(Tree t) {  if (t == NULL) {  printf("#");  }  else {  printf("%c", t->data);  if (t->left != NULL || t->right != NULL) {  printf("(");  printTree(t->left);  printf(",");  printTree(t->right);  printf(")");  }  }  }  int main() {  Tree t;  printf("请输入先序遍历序列（#表示空结点）：\n");  createTree(&t);  printf("广义表表示法：");  printTree(t);  printf("\n");  return 0;  }  #include <stdio.h>  #include <stdlib.h>  struct TreeNode {  int val;  struct TreeNode\* left;  struct TreeNode\* right;  };  // 先序遍历  void preorderTraversal(struct TreeNode\* root) {  if (root == NULL) {  return;  }  printf("%d ", root->val);  preorderTraversal(root->left);  preorderTraversal(root->right);  }  // 中序遍历  void inorderTraversal(struct TreeNode\* root) {  if (root == NULL) {  return;  }  inorderTraversal(root->left);  printf("%d ", root->val);  inorderTraversal(root->right);  }  // 后序遍历  void postorderTraversal(struct TreeNode\* root) {  if (root == NULL) {  return;  }  postorderTraversal(root->left);  postorderTraversal(root->right);  printf("%d ", root->val);  }  int main() {  // 构造二叉树  struct TreeNode\* root = (struct TreeNode\*)malloc(sizeof(struct TreeNode));  root->val = 1;  root->left = (struct TreeNode\*)malloc(sizeof(struct TreeNode));  root->left->val = 2;  root->left->left = NULL;  root->left->right = NULL;  root->right = (struct TreeNode\*)malloc(sizeof(struct TreeNode));  root->right->val = 3;  root->right->left = (struct TreeNode\*)malloc(sizeof(struct TreeNode));  root->right->left->val = 4;  root->right->left->left = NULL;  root->right->left->right = NULL;  root->right->right = (struct TreeNode\*)malloc(sizeof(struct TreeNode));  root->right->right->val = 5;  root->right->right->left = NULL;  root->right->right->right = NULL;  // 先序遍历  printf("Preorder traversal: ");  preorderTraversal(root);  printf("\n");  // 中序遍历  printf("Inorder traversal: ");  inorderTraversal(root);  printf("\n");  // 后序遍历  printf("Postorder traversal: ");  postorderTraversal(root);  printf("\n");  return 0;  }    #include <stdio.h>  #include <stdlib.h>  struct TreeNode {  int val;  struct TreeNode\* left;  struct TreeNode\* right;  };  // 求二叉树结点数  int countNodes(struct TreeNode\* root) {  if (root == NULL) {  return 0;  }  return 1 + countNodes(root->left) + countNodes(root->right);  }  // 求树的总结点数  int countTreeNodes(struct TreeNode\* root) {  if (root == NULL) {  return 0;  }  return 1 + countTreeNodes(root->left) + countTreeNodes(root->right);  }  // 求树的深度  int maxDepth(struct TreeNode\* root) {  if (root == NULL) {  return 0;  }  int leftDepth = maxDepth(root->left);  int rightDepth = maxDepth(root->right);  return (leftDepth > rightDepth) ? leftDepth + 1 : rightDepth + 1;  }  int main() {  // 构造一个二叉树  struct TreeNode\* root = (struct TreeNode\*)malloc(sizeof(struct TreeNode));  root->val = 1;  root->left = (struct TreeNode\*)malloc(sizeof(struct TreeNode));  root->left->val = 2;  root->left->left = NULL;  root->left->right = NULL;  root->right = (struct TreeNode\*)malloc(sizeof(struct TreeNode));  root->right->val = 3;  root->right->left = (struct TreeNode\*)malloc(sizeof(struct TreeNode));  root->right->left->val = 4;  root->right->left->left = NULL;  root->right->left->right = NULL;  root->right->right = (struct TreeNode\*)malloc(sizeof(struct TreeNode));  root->right->right->val = 5;  root->right->right->left = NULL;  root->right->right->right = NULL;  int nodeCount = countNodes(root);  int treeCount = countTreeNodes(root);  int depth = maxDepth(root);  printf("二叉树结点数：%d\n", nodeCount);  printf("树的总结点数：%d\n", treeCount);  printf("树的深度：%d\n", depth);  return 0;  }  #include <stdio.h>  int main() {  int choice;  printf("Welcome to the menu!\n");  printf("1. 选项 1\n");  printf("2. 选项 2\n");  printf("3. 选项n 3\n");  printf("4. 结束\n");  do {  printf("Enter your choice: ");  scanf\_s("%d", &choice);  switch (choice) {  case 1:  printf("你选择了 选项 1.\n");  break;  case 2:  printf("你选择了 选项 2.\n");  break;  case 3:  printf("你选择了 选项 3.\n");  break;  case 4:  printf("Goodbye!\n");  break;  default:  printf("Invalid choice. Please try again.\n");  break;  }  } while (choice != 4);  return 0;  }   1. 实验结果及分析 | | | | | | | |

说明：1. 实验周次：填写实际上课周，如第5-8周上课填“5-8”或第10周上课填“10”。

1. 实验报告各部分内容需详实填写，按实验指导书上的评分标准给出分数。
2. 实验类型参考实验类型说明文件。