**《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级二班 | 学生姓名 | 张文浩 |
| 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：80 | |
| 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   /\*树子系统\*/  #define \_CRT\_SECURE\_NO\_WARNINGS  #include <stdio.h>  #include <malloc.h>  #define MAX 100  int count = 0; /\*定义计算结点个数的变量\*/  typedef struct tnode  {  char data;  struct tnode\* lchild, \* rchild;  }BT;  BT\* CreateBTree()  {  BT\* t;  char ch;  scanf("%c", &ch);  getchar();  if (ch == '0')  t = NULL;  else  {  t = (BT\*)malloc(sizeof(BT));  t->data = ch;  printf("请输入%c结点的左孩子结点：", t->data);  t->lchild = CreateBTree();  printf("请输入%c结点的右孩子结点：", t->data);  t->rchild = CreateBTree();  }  return t;  }  void ShowBTree(BT\* T) /\*用广义表表示法显示二叉树\*/  {  if (T != NULL) /\*当二叉树非空时\*/  {  printf("%c", T->data); /\*输入该结点数据域\*/  if (T->lchild != NULL) /\*若其左子树非空\*/  {  printf("("); /\*输入左括号\*/  ShowBTree(T->lchild); /\*递归调用该函数输出其左子树各结点\*/  if (T->rchild != NULL) /\*若其右子树非空\*/  {  printf(","); /\*输出逗号\*/  ShowBTree(T->rchild); /\*递归调用该函数输出其右子树各结点\*/  }  printf(")");  }  else  if (T->rchild != NULL) /\*二叉树左子树为空，右子树不为空时\*/  {  printf("("); /\*输入左括号\*/  ShowBTree(T->lchild); /\*递归调用该函数输出其左子树各结点\*/  if (T->rchild != NULL) /\*若其右子树非空\*/  {  printf(","); /\*输出逗号\*/  ShowBTree(T->rchild); /\*递归调用该函数输出其右子树各结点\*/  }  printf(")");  }  }  }  void PreOrder(BT\* T) /\* 先序遍历二叉树T\*/  {  if (T == NULL) return; /\* 递归调用的结束条件\*/  else  {  printf("%c", T->data); /\* 输出结点的数据域\*/  PreOrder(T->lchild); /\* 先序递归遍历左子树\*/  PreOrder(T->rchild); /\* 先序递归遍历右子树\*/  }  }  void InOrder(BT\* T) /\* 中序遍历二叉树T\*/  {  if (T == NULL) return; /\* 递归调用的结束条件\*/  else  {  InOrder(T->lchild); /\* 中序递归遍历左子树\*/  printf("%c", T->data); /\* 输出结点的数据域\*/  InOrder(T->rchild); /\* 中序递归遍历右子树\*/  }  }  void PostOrder(BT\* T) /\* 后序遍历二叉树T\*/  {  if (T == NULL) return; /\* 递归调用的结束条件\*/  else  {  PostOrder(T->lchild); /\* 后序递归遍历左子树\*/  PostOrder(T->rchild); /\* 后序递归遍历右子树\*/  printf("%c", T->data); /\* 输出结点的数据域\*/  }  }  void LevelOrder(BT\* T) /\*按层次遍历二叉树T\*/  {  int f, r; /\*定义队头队尾指针\*/  BT\* p, \* q[MAX]; /\*定义循环队列，存放结点指针\*/  p = T;  if (p != NULL) /\*若二叉树非空，则根结点地址入队\*/  {  f = 1; q[f] = p; r = 2;  }  while (f != r) /\*队列不空时\*/  {  p = q[f];  printf("%c", p->data); /\*访问队首结点的数据域\*/  if (p->lchild != NULL) /\*将队首结点的左孩子入队\*/  {  q[r] = p->lchild; r = (r + 1) % MAX;  }  if (p->rchild != NULL) /\*将队首结点的右孩子入队\*/  {  q[r] = p->rchild; r = (r + 1) % MAX;  }  f = (f + 1) % MAX;  }  }  void Leafnum(BT\* T) /\*求二叉树叶子结点数\*/  {  if (T) /\*若树不为空\*/  {  if (T->lchild == NULL && T->rchild == NULL)  count++; /\*全局变量count为计数值，其初值为0\*/  Leafnum(T->lchild); /\*递归统计T的左子树叶子结点数\*/  Leafnum(T->rchild); /\*递归统计T的右子树叶子结点数\*/  }  }  void Nodenum(BT\* T)  {  if (T) /\*若树不为空\*/  {  count++; /\*全局变量count为计数值，其初值为0\*/  Nodenum(T->lchild); /\*递归统计T的左子树结点数\*/  Nodenum(T->rchild); /\*递归统计T的右子树结点数\*/  }  }  int TreeDepth(BT\* T) /\*求二叉树深度\*/  {  int ldep = 0, rdep = 0; /\*定义两个整型变量，用以存放左、右子树的深度\*/  if (T == NULL)  return 0;  else  {  ldep = TreeDepth(T->lchild); /\*递归统计T的左子树深度\*/  rdep = TreeDepth(T->rchild); /\*递归统计T的右子树深度\*/  if (ldep > rdep)  return ldep + 1;  else  return rdep + 1;  }  }  void MenuTree() /\*显示菜单子函数\*/  {  printf("\n 二叉树子系统");  printf("\n =================================================");  printf("\n| 1——建立一个新二叉树 |");  printf("\n| 2——广义表表示法显示 |");  printf("\n| 3——先序遍历 |");  printf("\n| 4——中序遍历 |");  printf("\n| 5——后序遍历 |");  printf("\n| 6——层次遍历 |");  printf("\n| 7——求叶子结点数目 |");  printf("\n| 8——求二叉树总结点数目 |");  printf("\n| 9——求树深度 |");  printf("\n| 0——返回 |");  printf("\n ================================================");  printf("\n请输入菜单号（0-9）:");  }  int main()  {  BT\* T = NULL;  char ch1, ch2, a;  ch1 = 'y';  while (ch1 == 'y' || ch1 == 'Y')  {  MenuTree();  scanf("%c", &ch2);  getchar();  switch (ch2)  {  case '1':  printf("请按先序序列输入二叉树的结点：\n");  printf("说明：输入结点后按回车（'0'表示后继结点为空）：\n");  printf("请输入根结点：");  T = CreateBTree();  printf("二叉树成功建立！"); break;  case '2':  printf("二叉树广义表表示法如下：");  ShowBTree(T); break;  case '3':  printf("二叉树的先序遍历序列为：");  PreOrder(T); break;  case '4':  printf("二叉树的中序遍历序列为：");  InOrder(T); break;  case '5':  printf("二叉树的后序遍历序列为：");  PostOrder(T); break;  case '6':  printf("二叉树的层次遍历序列为：");  LevelOrder(T); break;  case '7':  count = 0; Leafnum(T);  printf("该二叉树有%d个叶子。", count); break;  case '8':  count = 0; Nodenum(T);  printf("该二叉树共有%d个结点。", count); break;  case '9':  printf("该二叉树的深度是%d。", TreeDepth(T)); break;  case '0':  ch1 = 'n'; break;  default:  printf("输入有误，请输入0-9进行选择！");  }  if (ch2 != '0')  {  printf("\n按回车键继续，按任意键返回主菜单！\n");  a = getchar();  if (a != '\xA')  {  getchar(); ch1 = 'n';  }  }  }  }   1. 实验结果及分析   1.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。  21   1. 编写先序遍历、中序遍历、后序遍历程序   345  3.编写求二叉树结点数、树的总结点树和深度的程序。  6 78   1. 设计选择式菜单，以选择菜单的方式进行操作。 | | | | | | | |

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。