**新疆政法学院**

**数据结构与算法**

**实验报告**

班 级： 计算机科学与技术4班

学 号： 2124030160

姓 名： 孙久猛

指导老师： 张家琦

学 期： 2022-2023第一学期

|  |  |  |  |
| --- | --- | --- | --- |
| **实验题目** | 实验四：二叉树的前序遍历、中序遍历、后序遍历 | | |
| **实验时间** | **10:00** | **实验地点** | **东训0114** |
| **实验成绩** |  | **实验性质** | **□验证性 √设计性 □综合性** |
| 教师评价：  □算法/实验过程正确； □源程序/实验内容提交 □程序结构/实验步骤合理；  □实验结果正确； □语法、语义正确； □报告规范；  其他：  评价教师签名： | | | |
| 一、实验目的  1.理解二叉树的顺序存储、前序遍历、中序遍历、后序遍历的遍历思路；  2.初始化一个二叉树，并按照层次遍历依次输入二叉树的节点；  3.根据输入的数据创建二叉树；  4.根据创建好的二叉树依次进行前序遍历、中序遍历、后序遍历；  5.加深二叉树三种遍历方法的理解。 | | | |
| **二、实验项目内容（实验题目）**  （一）创建二叉树，按前序遍历求出遍历结果。  1.初始化一个二叉树，并按照层次遍历依次输入二叉树的节点；  3.根据输入的数据创建二叉树；  4.根据创建好的二叉树按前序遍历的方法求出遍历结果；  （二）在第一步的基础上，按中序遍历求遍历结果。  1.在第一步的基础上，针对已经创建好的二叉树按中序遍历的方法进行遍历；  2.输出遍历结果；  （三）在第一步的基础上，按后序遍历求遍历结果。  1.在第一步的基础上，针对已经创建好的二叉树按后序遍历的方法进行遍历；  2.输出遍历结果；  **三、源程序（实验步骤/实验过程/算法）**  //Demo\_01.c  #include<stdio.h>  #define OK 1  #define ERROR 0  #define TRUE 1  #define FALSE 0  typedef int Status;  #if CHAR  typedef char TElemType;  TElemType Nil=' ';  #else  typedef int TElemType;  TElemType Nil=0; /\* 设整型以0为空 \*/  #endif  #define MAX\_TREE\_SIZE 100 /\* 二叉树的最大结点数 \*/  typedef TElemType SqBiTree[MAX\_TREE\_SIZE]; /\* 0号单元存储根结点 \*/  Status InitBiTree(SqBiTree T)  {  int i;  for(i=0;i<MAX\_TREE\_SIZE;i++)  T[i]=Nil;  return OK;  }  Status CreateBiTree(SqBiTree T)  { /\* 按层序次序输入二叉树中结点的值(字符型或整型), 构造顺序存储的二叉树T \*/  int i=0;  #if CHAR  #else  printf("请按层序输入结点的值(整型)，0表示空结点，输999结束。结点数≤%d:\n",MAX\_TREE\_SIZE);  while(1)  {  scanf("%d",&T[i]);  if(T[i]==999)  break;  if(i!=0&&T[(i+1)/2-1]==Nil&&T[i]!=Nil)  {  printf("出现无双亲的非根结点%d\n",T[i]);  return FALSE;  }  i++;  }  while(i<MAX\_TREE\_SIZE)  {  T[i]=Nil; /\* 将空赋值给T的后面的结点 \*/  i++;  }  #endif  return OK;  }  Status BiTreeEmpty(SqBiTree T)  { /\* 初始条件: 二叉树T存在 \*/  /\* 操作结果: 若T为空二叉树,则返回TRUE,否则FALSE \*/  if(T[0]==Nil) /\* 根结点为空,则树空 \*/  return TRUE;  else  return FALSE;  }  Status(\*VisitFunc)(TElemType);  void InTraverse(SqBiTree T,int e)  { /\* InOrderTraverse()调用 \*/  if(T[2\*e+1]!=Nil) /\* 左子树不空 \*/  InTraverse(T,2\*e+1);  VisitFunc(T[e]);  if(T[2\*e+2]!=Nil) /\* 右子树不空 \*/  InTraverse(T,2\*e+2);  }  Status InOrderTraverse(SqBiTree T,Status(\*Visit)(TElemType))  { /\* 初始条件: 二叉树存在,Visit是对结点操作的应用函数 \*/  /\* 操作结果: 中序遍历T,对每个结点调用函数Visit一次且仅一次。 \*/  /\* 一旦Visit()失败,则操作失败 \*/  VisitFunc=Visit;  if(!BiTreeEmpty(T))  InTraverse(T,0);  printf("\n");  return OK;  }  void PostTraverse(SqBiTree T,int e)  { /\* PostOrderTraverse()调用 \*/  if(T[2\*e+1]!=Nil) /\* 左子树不空 \*/  PostTraverse(T,2\*e+1);  if(T[2\*e+2]!=Nil) /\* 右子树不空 \*/  PostTraverse(T,2\*e+2);  VisitFunc(T[e]);  }  Status PostOrderTraverse(SqBiTree T,Status(\*Visit)(TElemType))  { /\* 初始条件: 二叉树T存在,Visit是对结点操作的应用函数 \*/  /\* 操作结果: 后序遍历T,对每个结点调用函数Visit一次且仅一次。 \*/  /\* 一旦Visit()失败,则操作失败 \*/  VisitFunc=Visit;  if(!BiTreeEmpty(T))  PostTraverse(T,0);  printf("\n");  return OK;  }  void PreTraverse(SqBiTree T,int e)  { /\* PostOrderTraverse()调用 \*/  VisitFunc(T[e]);  if(T[2\*e+1]!=Nil) /\* 左子树不空 \*/  PreTraverse(T,2\*e+1);  if(T[2\*e+2]!=Nil) /\* 右子树不空 \*/  PreTraverse(T,2\*e+2);  }  Status PreOrderTraverse(SqBiTree T,Status(\*Visit)(TElemType))  { /\* 初始条件: 二叉树T存在,Visit是对结点操作的应用函数 \*/  /\* 操作结果: 前序遍历T,对每个结点调用函数Visit一次且仅一次。 \*/  /\* 一旦Visit()失败,则操作失败 \*/  VisitFunc=Visit;  if(!BiTreeEmpty(T))  PreTraverse(T,0);  printf("\n");  return OK;  }  Status visit(TElemType e)  {  printf("%d ",e);  return OK;  }  int main()  {  int n;  SqBiTree T;  InitBiTree(T);  CreateBiTree(T);  printf("前序遍历二叉树:\n");  PreOrderTraverse(T,visit);  printf("中序遍历二叉树:\n");  InOrderTraverse(T,visit);  printf("后序遍历二叉树:\n");  PostOrderTraverse(T,visit);  return 0;  }  **四、运行结果**  **//Doem\_04.exe**  //Picture\_01    //Picture\_02 | | | |

日 期： 2022年10月19日