

东莞城市学院人工智能学院

School of Artificial Intelligence

《数据结构》

实验报告

| 学号 | 202435710252 |
| --- | --- |
| 姓名 | 钟润柯 |
| 专业、班级 | 2024级计科（专升本）2班 |
| 课程名称 | 《数据结构》 |
| 指导教师 | 李言一 |
| 学期 | 2024—2025第1学期 |
| 实验次序 | 实验五 |
| 上机时间 | 2024-11-20 周三 第5-6节 |
| 上机地点 | 3B416 |
| 分数 |  |

二○二三——二○二四学年 第一学期

实验五 二叉树的基本操作

# 实验课时

课内：2课时

# 实验类型

操作性实验

# 实验目的

1.熟练掌握二叉树的二叉链表存储结构。

2.掌握二叉树的非线性和递归性特点。

3.熟练掌握二叉树的递归遍历操作的实现方法，掌握二叉树的非递归遍历操作的实现。

4.加深对二叉树结构和性质的理解。

5.逐步培养解决实际问题的编程能力。

# 实验内容和步骤

## 一、源代码

（1）二叉树定义和创建及主函数的源码

|  |
| --- |
| #include <stdio.h>  #include <stdlib.h>  typedef char ElemType;  typedef struct BinTreeNode {  ElemType data;  struct BinTreeNode \*lchild;  struct BinTreeNode \*rchild;  } BinTreeNode, \*BinTree;  void CreateTree(BinTree \*T) {  ElemType ch;  scanf("%c", &ch);  if (ch == '#') {  \*T = NULL;  } else {  \*T = (BinTree) malloc(sizeof(BinTreeNode));  if (!\*T) {  return;  }  (\*T) -> data= ch;  CreateTree(&(\*T)->lchild);  CreateTree(&(\*T)->rchild);  }  }  int main() {      // BinTreeNode bin;      BinTree root = NULL;      CreateTree(&root);      // ABD###CE##F##      printf("前序遍历结果为 :");      PreOrderTraverse(root);      printf("\n中序遍历结果为 :");      InOrderTraverse(root);      printf("\n后序遍历结果为 :");      PostOrderTraverse(root);      int n = LeafCount(root);      printf("\n叶子数为：%d", n);      int m = AllNodeCount(root);      printf("\n总节点数为：%d", m);  } |

（2）二叉树前序遍历的源码：

|  |
| --- |
| void PreOrderTraverse(BinTree T) {      if (T == NULL) {          return;      }      printf("%c ", T->data);      PreOrderTraverse(T->lchild);      PreOrderTraverse(T->rchild);  } |

（3）二叉树中序遍历的源码：

|  |
| --- |
| void InOrderTraverse(BinTree T) {      if (T == NULL) {          return;      }      InOrderTraverse(T->lchild);      printf("%c ", T->data);      InOrderTraverse(T->rchild);  } |

（4）二叉树后序遍历的源码

|  |
| --- |
| void PostOrderTraverse(BinTree T) {      if (T == NULL) {          return;      }      PostOrderTraverse(T->lchild);      PostOrderTraverse(T->rchild);      printf("%c ", T->data);  } |

（5）二叉树计算叶子结点数的源码

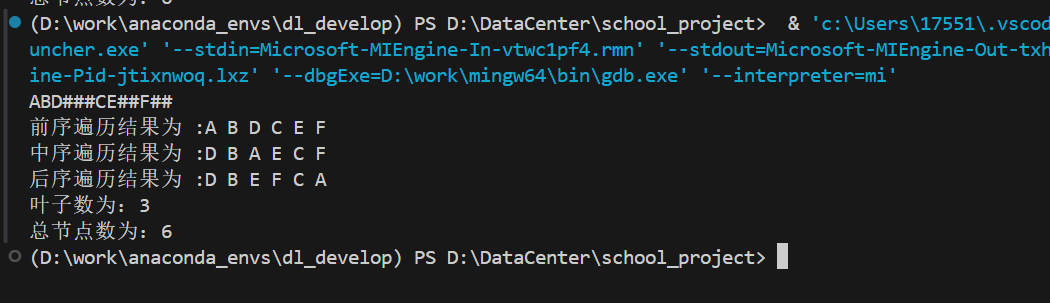
|  |
| --- |
| int LeafCount(BinTree T) {      if (T==NULL) {          return 0;      }      if (T->lchild==NULL && T->rchild == NULL) {          return 1;      } else {          return LeafCount(T->lchild)+LeafCount(T->rchild);      }  } |

（6）二叉树计算结点总数的源码

|  |
| --- |
| int AllNodeCount(BinTree T) {      if (T==NULL) {          return 0;      } else {          return AllNodeCount(T->lchild)+AllNodeCount(T->rchild)+1;      }  } |

## 二、程序运行结果截图

（1）二叉树链表定义、初始化、前序、中序、后序遍历和计算叶子及结点总数运行结果：



## 三、实验总结

通过本次实验，我深刻理解了二叉树的链式存储结构和递归遍历方法。在实现过程中，我遇到了一些问题，如内存分配失败和输入格式错误，通过调试和查阅资料，我学会了如何解决这些问题。此外，我还认识到递归算法在二叉树操作中的重要性，以及如何通过递归方法简化问题的解决。