上海工程技术大学

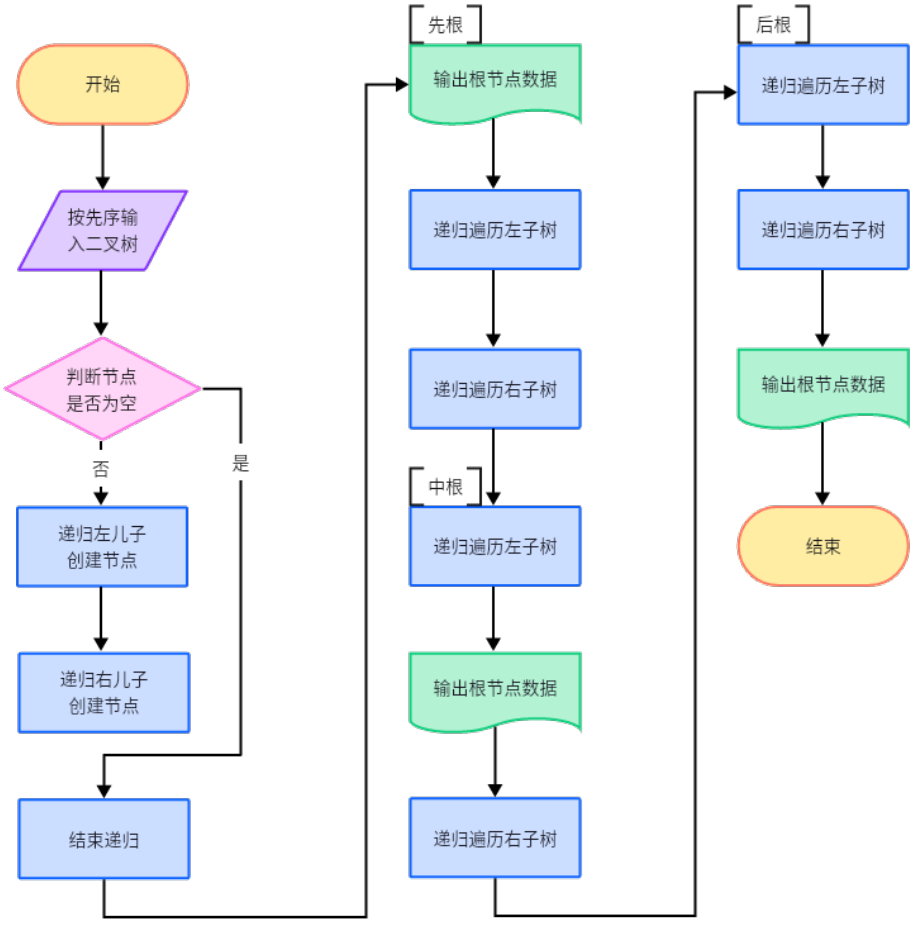
实 验 报 告

实验二　二叉树的应用

1. 题目 1

设计一个程序，由给定的二叉树先序序列，建立其二叉链表存储结构，并求出二叉树的中序序列和后序序列。

1.1. 流程图：



1.2. 代码

```
#include <iostream>
using namespace std;

typedef char ElemType;
typedef struct node {
    ElemType data;
    struct node* LChild, * RChild;
}BNode,*BTree;

void InitBTree(BTree& BT) {
    BT = NULL;
}

void CreateBTree(BTree& BT) {
    ElemType data;
    cin >> data;
    if (data == '#')
        BT = NULL;
    else {
        BT = (BTree)malloc(sizeof(BNode));
        memset(BT,0x00f,sizeof(BT));
        BT->data = data;
        CreateBTree(BT->LChild);
        CreateBTree(BT->RChild);
    }
}

void PreOrder(BTree p) {
    if (p != NULL) {
        cout << p->data << "\t";
        PreOrder(p->LChild);
        PreOrder(p->RChild);
    }
}

void InOrder(BTree p) {
    if (p != NULL) {
        InOrder(p->LChild);
        cout << p->data << "\t";
        InOrder(p->RChild);
    }
}
```

```
void PostOrder(BTree p) {
    if (p != NULL) {
        PostOrder(p->LChild);
        PostOrder(p->RChild);
        cout << p->data << "\t";
    }
}

int main() {
    BTree p;
    InitBTree(p);
    cout << "Input the preorder of the BTree:";
    CreateBTree(p);
    cout << "The preorder of the BTree:\n";
    PreOrder(p);
    cout << endl;
    cout << "The inorder of the BTree:\n";
    InOrder(p);
    cout << endl;
    cout << "The postorder of the BTree:\n";
    PostOrder(p);
    cout << endl;
    return 0;
}
```

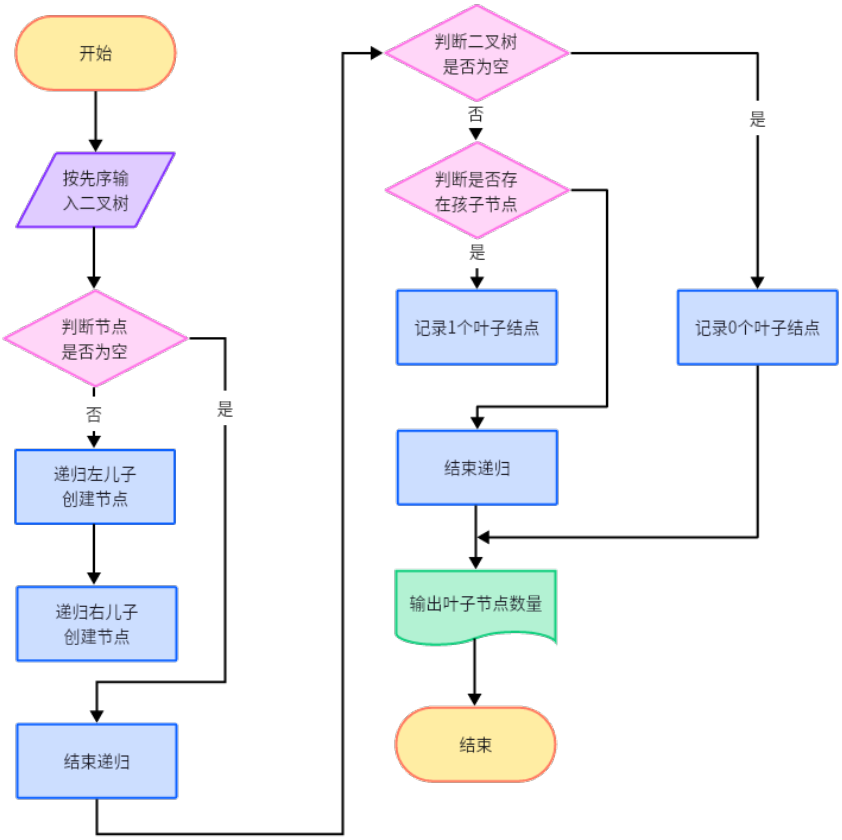
1.3. 结果

```
Input the preorder of the BTree:AB#CD####
The preorder of the BTree:
A       B       C       D
The inorder of the BTree:
B       D       C       A
The postorder of the BTree:
D       C       B       A
```

2. 题目 2

假设二叉树采用二叉链存储结构，编写一个算法，求出二叉树中的叶子结点数。并设计主函数调用上述算法。

2.1. 流程图：



2.2. 代码

```
#include<stdio.h>
#include<stdlib.h>

typedef struct node{
    char data ;
    struct node * lchild;
    struct node * rchild;
}BiTree;

BiTree * CreatTree(){
    BiTree *t;
    char ch ;
    ch = getchar();
    if (ch != '#'){
        t = (BiTree *)malloc(sizeof(BiTree));
        t ->data = ch ;
        t->lchild = CreatTree();
        t->rchild = CreatTree();
    }
    else{
```

```
        t=NULL;
    }
    return t;
}

int Count(BiTree * top){
    if(top == NULL){
        return 0;
    }
    else if ((top->lchild==NULL) && (top->rchild==NULL)){
        return 1;
    }
    else{
        return Count(top->lchild)+Count(top->rchild);
    }
}

void Preorder(BiTree * top ){
    if(top != NULL){
        printf("%c ",top->data);
        Preorder(top->lchild);
        Preorder(top->rchild);
    }
}

int main(){
    BiTree * top = NULL;
    printf("用先根遍历输入树：");
    top = CreatTree();
    printf("先根遍历结果：");
    Preorder(top);
    putchar('\n');
    printf("叶子节点的个数：  %d\n",Count(top));
    return 0;
}
```

2.3. 结果

```
用先根遍历输入树：AB#CD####
先根遍历结果：A B C D
叶子节点的个数：  1
```

3. 实验小结

我首先了解了二叉树的基本概念和二叉链表的存储方式。然后，根据给定的先序序列，

我按照递归的思路建立了二叉链表，并分别求出了中序序列和后序序列。在此过程中，我深刻体会到了递归算法的实现方法及其递归调用的顺序问题，也发现使用递归算法的过程中会大大增加时间复杂度。通过对整个二叉树进行遍历，在遍历到叶子结点时进行计数，最终得到了正确的结果。在实现这个算法的过程中，我对二叉树的遍历算法有了更深入的理解。

通过本次实验，我掌握了二叉树的基本概念、递归算法的实现方法，以及二叉树的遍历算法等知识点。