# 实验4 二叉树及其应用

**实验日期： 2021 年 5 月 6 日---- 5 月 6 日 报告日期：2021 年 5月 6日**

### 一、实验目的

**1、掌握二叉树的基本特性**

**2、掌握二叉树的先序、中序、后序的递归遍历算法**

**3、通过求二叉树的深度、叶子结点数等算法，理解二叉树的基本特性**

### 二、实验预习

**复习以下概念**

**1、二叉树：树中节点的度不大于2的有序树**

**2、递归遍历：包含递归前序遍历、递归中序遍历、递归后序遍历三种方法。字面意思：通过递归的方式挨个遍历各个元素结点**

### 三、实验内容和要求

**1、阅读并运行下面程序，根据输入写出运行结果，并画出二叉树的形态。**

#include<stdio.h>

#include<malloc.h>

**typedef struct BTNode{ /\*结点结构声明\*/**

**char data ; /\*结点数据\*/**

**struct BTNode \*lchild;**

**struct BTNode \*rchild ; /\*指针\*/**

**}\*BiTree;**

int createBiTree(BiTree &t){ /\* 先序遍历创建二叉树\*/

char s;

BiTree q;

s=getchar();

if(s=='.'){t=NULL; return 1;}

q=(BiTree)malloc(sizeof(struct BTNode));

if(q==NULL){printf("Memory alloc failure!"); return 0;}

q->data=s;

t=q;

createBiTree(q->lchild); /\*递归建立左子树\*/

createBiTree(q->rchild); /\*递归建立右子树\*/

return 1;

}

void PreOrder(BiTree p){ /\* 先序遍历二叉树\*/

if ( p!= NULL ) {

printf("%c", p->data);

PreOrder( p->lchild ) ;

PreOrder( p->rchild) ;

}

}

void InOrder(BiTree p){ /\* 中序遍历二叉树\*/

if( p!= NULL ) {

InOrder( p->lchild ) ;

printf("%c", p->data);

InOrder( p->rchild) ;

}

}

void PostOrder(BiTree p){ /\* 后序遍历二叉树\*/

if ( p!= NULL ) {

PostOrder( p->lchild ) ;

PostOrder( p->rchild) ;

printf("%c", p->data);

}

}

void release(BiTree t){ /\*释放二叉树空间\*/

if(t!=NULL){

release(t->lchild);

release(t->rchild);

free(t);

}

}

/\*

int nodenum(BiTree p) //求二叉树结点数

{int n;

return n;

}

int leafnum(BiTree p) //求二叉树叶子结点数

{int n0;

if(p==NULL) n0=0;

else

if(p->lchild==NULL && p->rchild==NULL)

n0= ;

else

n0=leafnum(p->lchild)+ ;

return ;

}

int depth(BiTree p) //求二叉树深度

{int d1,d2;

if(p==NULL) return 0;

else

{d1= ;

d2= ;

return ;

}

}

\*/

int main(){

BiTree t=NULL;

printf("\nplease input data:(用.表示空树)");

createBiTree(t);

printf("\n\nPreOrder the tree is:");

PreOrder(t);

printf("\n\nInOrder the tree is:");

InOrder(t);

printf("\n\nPostOrder the tree is:");

PostOrder(t);

/\*

printf("\n该二叉树有%d个结点\n",nodenum(t));

printf("\n该二叉树有%d个叶子结点\n",leafnum(t));

printf("\n该二叉树的深度为%d\n",depth(t));

\*/

release(t);

return 0;

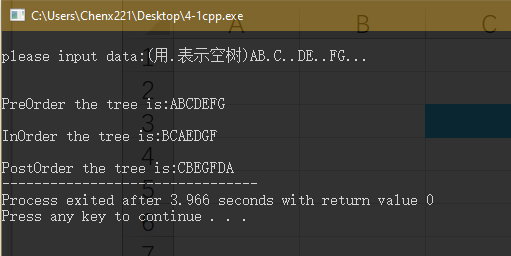
}

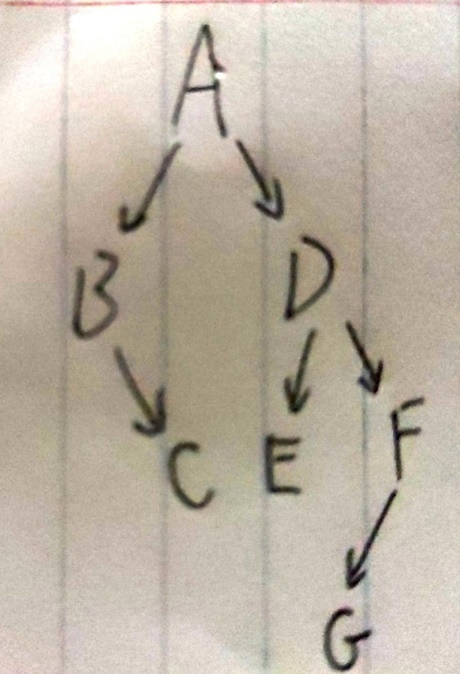
**运行程序**

**输入：**

**AB.C..DE..FG...**

**运行结果：**





/\*阅读并运行下面程序，根据输入写出运行结果，并画出二叉树的形态。

文件名：\*\_实验4-1.cpp

by Chenx221 (https://chenx221.cyou)

\*/

#include<stdio.h>

#include<malloc.h>

typedef struct BTNode {       /\*结点结构声明\*/

    char data;               /\*结点数据\*/

    struct BTNode \*lchild;

    struct BTNode \*rchild;  /\*指针\*/

} \*BiTree;

int createBiTree(BiTree &t) { /\* !先序遍历创建二叉树\*/

    char s;

    BiTree q;

    s = getchar();

    if (s == '.') { //空树

        t = NULL;

        return 1;

    }

    q = (BiTree) malloc(sizeof(struct BTNode));

    if (q == NULL) { //内存分配失败

        printf("Memory alloc failure!");

        return 0;

    }

    q->data = s;

    t = q;

    createBiTree(q->lchild); /\*递归建立左子树\*/

    createBiTree(q->rchild); /\*递归建立右子树\*/

    return 1;

}

void PreOrder(BiTree p) {  /\* 先序遍历二叉树\*/

    if (p != NULL) {

        printf("%c", p->data);

        PreOrder(p->lchild);

        PreOrder(p->rchild);

    }

}

void InOrder(BiTree p) {  /\* 中序遍历二叉树\*/

    if (p != NULL) {

        InOrder(p->lchild);

        printf("%c", p->data);

        InOrder(p->rchild);

    }

}

void PostOrder(BiTree p) {  /\* 后序遍历二叉树\*/

    if (p != NULL) {

        PostOrder(p->lchild);

        PostOrder(p->rchild);

        printf("%c", p->data);

    }

}

void release(BiTree t) { /\*释放二叉树空间\*/

    if (t != NULL) {

        release(t->lchild);

        release(t->rchild);

        free(t);

    }

}

/\*

int nodenum(BiTree p) {  //求二叉树结点数

    int n;

    return n;

}

int leafnum(BiTree p)  //求二叉树叶子结点数

{

    int n0;

    if (p == NULL) n0 = 0;

    else if (p->lchild == NULL && p->rchild == NULL)

        n0 =;

    else

        n0 = leafnum(p->lchild) +;

    return;

}

int depth(BiTree p)   //求二叉树深度

{

    int d1, d2;

    if (p == NULL) return 0;

    else {

        d1 =;

        d2 =;

        return;

    }

}

\*/

int main() {

    BiTree t = NULL;

    printf("\nplease input data:(用.表示空树)");

    createBiTree(t); //创建二叉树，先序遍历传入数据

    printf("\n\nPreOrder the tree is:");

    PreOrder(t); //先序输出

    printf("\n\nInOrder the tree is:");

    InOrder(t); //中

    printf("\n\nPostOrder the tree is:");

    PostOrder(t); //后

    /\*

      printf("\n该二叉树有%d个结点\n",nodenum(t));

       printf("\n该二叉树有%d个叶子结点\n",leafnum(t));

   printf("\n该二叉树的深度为%d\n",depth(t));

   \*/

    release(t); //释放

    return 0;

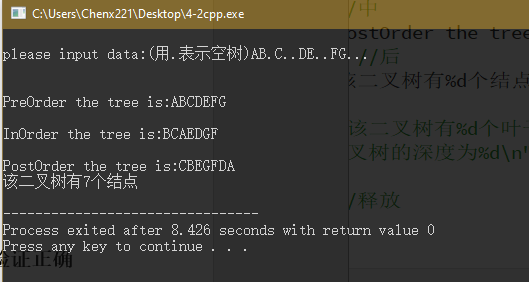
}

**2、在上题中补充完成求二叉树中求结点总数算法，并在主函数中补充相应的调用验证正确性。**

**输入：**

**AB.C..DE..FG...**

**运行结果：**



/\*在上题中补充完成求二叉树中求结点总数算法，并在主函数中补充相应的调用验证正确性

文件名：\*\_实验4-2.cpp

by Chenx221 (https://chenx221.cyou)

\*/

#include<stdio.h>

#include<malloc.h>

typedef struct BTNode {       /\*结点结构声明\*/

    char data;               /\*结点数据\*/

    struct BTNode \*lchild;

    struct BTNode \*rchild;  /\*指针\*/

} \*BiTree;

int createBiTree(BiTree &t) { /\* !先序遍历创建二叉树\*/

    char s;

    BiTree q;

    s = getchar();

    if (s == '.') { //空树

        t = NULL;

        return 1;

    }

    q = (BiTree) malloc(sizeof(struct BTNode));

    if (q == NULL) { //内存分配失败

        printf("Memory alloc failure!");

        return 0;

    }

    q->data = s;

    t = q;

    createBiTree(q->lchild); /\*递归建立左子树\*/

    createBiTree(q->rchild); /\*递归建立右子树\*/

    return 1;

}

void PreOrder(BiTree p) {  /\* 先序遍历二叉树\*/

    if (p != NULL) {

        printf("%c", p->data);

        PreOrder(p->lchild);

        PreOrder(p->rchild);

    }

}

void InOrder(BiTree p) {  /\* 中序遍历二叉树\*/

    if (p != NULL) {

        InOrder(p->lchild);

        printf("%c", p->data);

        InOrder(p->rchild);

    }

}

void PostOrder(BiTree p) {  /\* 后序遍历二叉树\*/

    if (p != NULL) {

        PostOrder(p->lchild);

        PostOrder(p->rchild);

        printf("%c", p->data);

    }

}

void release(BiTree t) { /\*释放二叉树空间\*/

    if (t != NULL) {

        release(t->lchild);

        release(t->rchild);

        free(t);

    }

}

int nodenum(BiTree p) {  //求二叉树结点数

    int n;

    if (p != NULL) {

        n = 1;

        n += nodenum(p->lchild);

        n += nodenum(p->rchild);

    } else n = 0;

    return n;

}

/\*

int leafnum(BiTree p)  //求二叉树叶子结点数

{int n0;

 if(p==NULL) n0=0; //空树

 else

    if(p->lchild==NULL && p->rchild==NULL) //无左右子 即为叶子结点

        n0=n0+1;

    else

        n0=leafnum(p->lchild)+               ;

 return  n0;

}

int depth(BiTree p)   //求二叉树深度

{int d1,d2;

 if(p==NULL) return 0;

 else

  {d1=       ;

   d2=       ;

   return        ;

  }

}

\*/

int main() {

    BiTree t = NULL;

    printf("\nplease input data:(用.表示空树)");

    createBiTree(t); //创建二叉树，先序遍历传入数据

    printf("\n\nPreOrder the tree is:");

    PreOrder(t); //先序输出

    printf("\n\nInOrder the tree is:");

    InOrder(t); //中

    printf("\n\nPostOrder the tree is:");

    PostOrder(t); //后

    printf("\n该二叉树有%d个结点\n", nodenum(t));

    /\*

       printf("\n该二叉树有%d个叶子结点\n",leafnum(t));

   printf("\n该二叉树的深度为%d\n",depth(t));

   \*/

    release(t); //释放

    return 0;

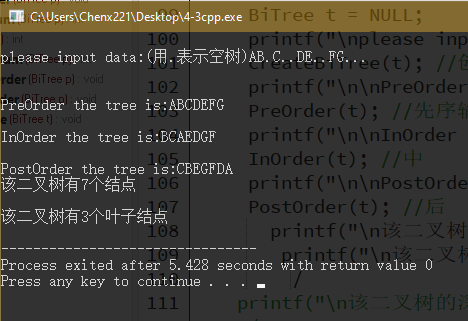
}

**3、在上题中补充完成求二叉树中求叶子结点总数算法，并在主函数中补充相应的调用验证正确性。**

**输入：**

**AB.C..DE..FG...**

**运行结果：**



/\*在上题中补充完成求二叉树中求叶子结点总数算法，并在主函数中补充相应的调用验证正确性

文件名：\*\_实验4-3.cpp

by Chenx221 (https://chenx221.cyou)

\*/

#include<stdio.h>

#include<malloc.h>

typedef struct BTNode {       /\*结点结构声明\*/

    char data;               /\*结点数据\*/

    struct BTNode \*lchild;

    struct BTNode \*rchild;  /\*指针\*/

} \*BiTree;

int createBiTree(BiTree &t) { /\* !先序遍历创建二叉树\*/

    char s;

    BiTree q;

    s = getchar();

    if (s == '.') { //空树

        t = NULL;

        return 1;

    }

    q = (BiTree) malloc(sizeof(struct BTNode));

    if (q == NULL) { //内存分配失败

        printf("Memory alloc failure!");

        return 0;

    }

    q->data = s;

    t = q;

    createBiTree(q->lchild); /\*递归建立左子树\*/

    createBiTree(q->rchild); /\*递归建立右子树\*/

    return 1;

}

void PreOrder(BiTree p) {  /\* 先序遍历二叉树\*/

    if (p != NULL) {

        printf("%c", p->data);

        PreOrder(p->lchild);

        PreOrder(p->rchild);

    }

}

void InOrder(BiTree p) {  /\* 中序遍历二叉树\*/

    if (p != NULL) {

        InOrder(p->lchild);

        printf("%c", p->data);

        InOrder(p->rchild);

    }

}

void PostOrder(BiTree p) {  /\* 后序遍历二叉树\*/

    if (p != NULL) {

        PostOrder(p->lchild);

        PostOrder(p->rchild);

        printf("%c", p->data);

    }

}

void release(BiTree t) { /\*释放二叉树空间\*/

    if (t != NULL) {

        release(t->lchild);

        release(t->rchild);

        free(t);

    }

}

int nodenum(BiTree p) {  //求二叉树结点数

    int n;

    if (p != NULL) {

        n = 1;

        n += nodenum(p->lchild);

        n += nodenum(p->rchild);

    } else n = 0;

    return n;

}

int leafnum(BiTree p) { //求二叉树叶子结点数

    int n0;

    if (p == NULL) n0 = 0; //空树

    else if (p->lchild == NULL && p->rchild == NULL) //无左右子 即为叶子结点

        n0 = 1;

    else

        n0 = leafnum(p->lchild) + leafnum(p->rchild);

    return n0;

}

/\*

int depth(BiTree p)   //求二叉树深度

{int d1,d2;

 if(p==NULL) return 0;

 else

  {d1=       ;

   d2=       ;

   return        ;

  }

}

\*/

int main() {

    BiTree t = NULL;

    printf("\nplease input data:(用.表示空树)");

    createBiTree(t); //创建二叉树，先序遍历传入数据

    printf("\n\nPreOrder the tree is:");

    PreOrder(t); //先序输出

    printf("\n\nInOrder the tree is:");

    InOrder(t); //中

    printf("\n\nPostOrder the tree is:");

    PostOrder(t); //后

    printf("\n该二叉树有%d个结点\n", nodenum(t));

    printf("\n该二叉树有%d个叶子结点\n", leafnum(t));

    /\*

printf("\n该二叉树的深度为%d\n",depth(t));

\*/

    release(t); //释放

    return 0;

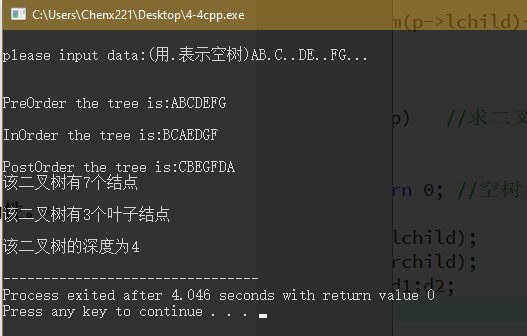
}

**4、在上题中补充完成求二叉树深度算法，并在主函数中补充相应的调用验证正确性。**

**输入：**

**AB.C..DE..FG...**

**运行结果：**



/\*在上题中补充完成求二叉树深度算法，并在主函数中补充相应的调用验证正确性。

文件名：\*\_实验4-4.cpp

by Chenx221 (https://chenx221.cyou)

\*/

#include<stdio.h>

#include<malloc.h>

typedef struct BTNode {       /\*结点结构声明\*/

    char data;               /\*结点数据\*/

    struct BTNode \*lchild;

    struct BTNode \*rchild;  /\*指针\*/

} \*BiTree;

int createBiTree(BiTree &t) { /\* !先序遍历创建二叉树\*/

    char s;

    BiTree q;

    s = getchar();

    if (s == '.') { //空树

        t = NULL;

        return 1;

    }

    q = (BiTree) malloc(sizeof(struct BTNode));

    if (q == NULL) { //内存分配失败

        printf("Memory alloc failure!");

        return 0;

    }

    q->data = s;

    t = q;

    createBiTree(q->lchild); /\*递归建立左子树\*/

    createBiTree(q->rchild); /\*递归建立右子树\*/

    return 1;

}

void PreOrder(BiTree p) {  /\* 先序遍历二叉树\*/

    if (p != NULL) {

        printf("%c", p->data);

        PreOrder(p->lchild);

        PreOrder(p->rchild);

    }

}

void InOrder(BiTree p) {  /\* 中序遍历二叉树\*/

    if (p != NULL) {

        InOrder(p->lchild);

        printf("%c", p->data);

        InOrder(p->rchild);

    }

}

void PostOrder(BiTree p) {  /\* 后序遍历二叉树\*/

    if (p != NULL) {

        PostOrder(p->lchild);

        PostOrder(p->rchild);

        printf("%c", p->data);

    }

}

void release(BiTree t) { /\*释放二叉树空间\*/

    if (t != NULL) {

        release(t->lchild);

        release(t->rchild);

        free(t);

    }

}

int nodenum(BiTree p) {  //求二叉树结点数

    int n;

    if (p != NULL) {

        n = 1;

        n += nodenum(p->lchild);

        n += nodenum(p->rchild);

    } else n = 0;

    return n;

}

int leafnum(BiTree p) { //求二叉树叶子结点数

    int n0;

    if (p == NULL) n0 = 0; //空树

    else if (p->lchild == NULL && p->rchild == NULL) //无左右子 即为叶子结点

        n0 = 1;

    else

        n0 = leafnum(p->lchild) + leafnum(p->rchild);

    return n0;

}

int depth(BiTree p) {  //求二叉树深度

    int d1, d2;

    if (p == NULL) return 0; //空树当然0深度

    else {

        d1 = 1 + depth(p->lchild); //不便解释

        d2 = 1 + depth(p->rchild); //不便解释

        return d1 > d2 ? d1 : d2;

    }

}

int main() {

    BiTree t = NULL;

    printf("\nplease input data:(用.表示空树)");

    createBiTree(t); //创建二叉树，先序遍历传入数据

    printf("\n\nPreOrder the tree is:");

    PreOrder(t); //先序输出

    printf("\n\nInOrder the tree is:");

    InOrder(t); //中

    printf("\n\nPostOrder the tree is:");

    PostOrder(t); //后

    printf("\n该二叉树有%d个结点\n", nodenum(t));

    printf("\n该二叉树有%d个叶子结点\n", leafnum(t));

    printf("\n该二叉树的深度为%d\n", depth(t));

    release(t); //释放

    return 0;

}

### 四、实验小结

**最开始没注意数据是以何种顺序输入的，阅读提供的源码 createBitree()函数部分才了解到是先序遍历输入数据**

**写求结点数部分代码时，考虑到本实验主要与递归有关，故使用了递归的方式计算结点数**

**叶子结点数没什么可以讲的**

**求深度部分的代码，一开始写成了**

  d1 = depth(p->lchild);

         d2 = depth(p->rchild);

运行时才注意到结果全成了0…