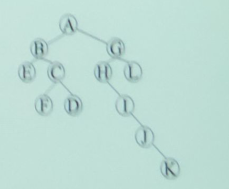
二叉树作业



先序 ABECFDGHLIJK

中序 EBFCDAHIJKGL

后序 EFDCBKJIHLGA

交换左右子树

算法

void Exchange(Tree \*bt) //交换左右子树

{

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

;

else

{

Tree \*temp=bt->lchild; //用temp暂时存储左孩子数值

bt->lchild=bt->rchild;

bt->rchild=temp;

}

if(bt->lchild) //分别判断左右孩子是否为空

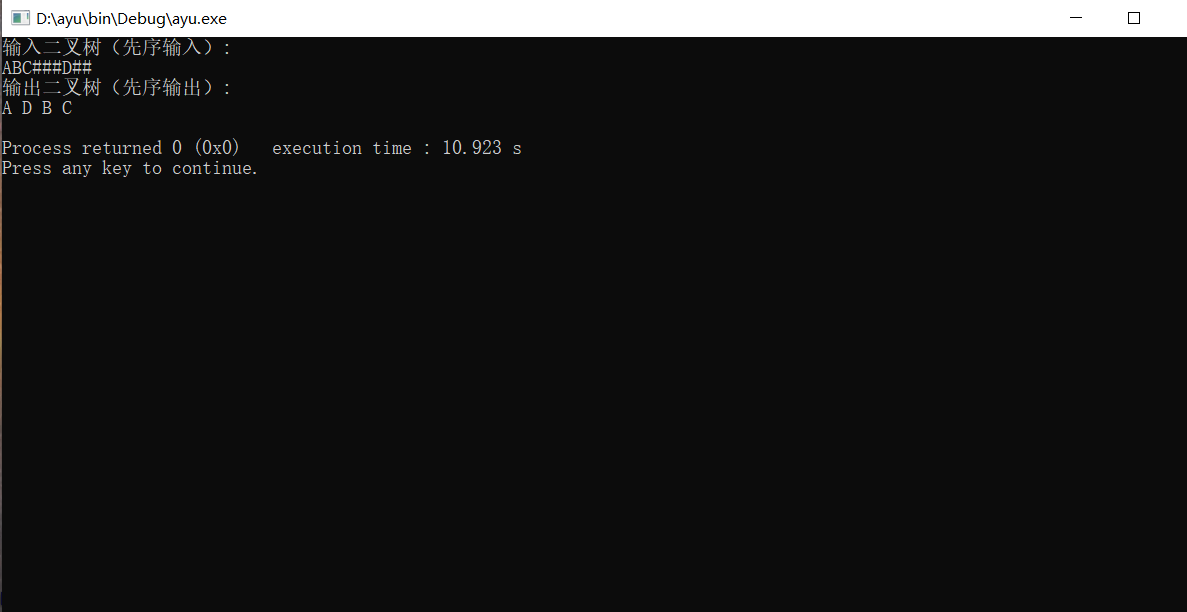
Exchange(bt->lchild); //递归继续寻找可交换节点

if(bt->rchild)

Exchange(bt->rchild);//递归继续寻找可交换节点

}

程序运行截图



统计二叉树中叶子结点个数

算法

int Leaf(Tree \*bt,int count)

{

if(bt)

{

if(bt->lchild==NULL && bt->rchild==NULL) //叶子结点左右孩子为空

{

count++;

}

count=Leaf(bt->lchild,count); //递归判断

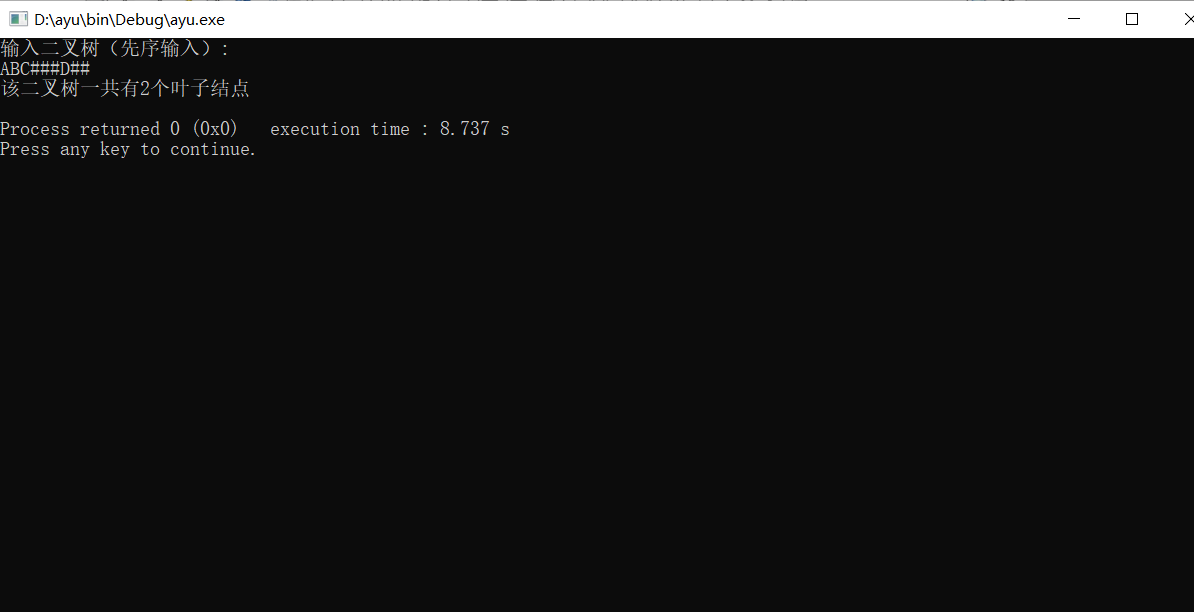
count=Leaf(bt->rchild,count); //递归判断

}

return count;

}

运行截图



二叉树深度计算

算法

int deep(Tree \*bt)

{

int ld=0,rd=0; //ld为左侧深度 rd为右侧

if(bt)

{

ld=deep(bt->lchild)+1; //每递归一次，ld+1

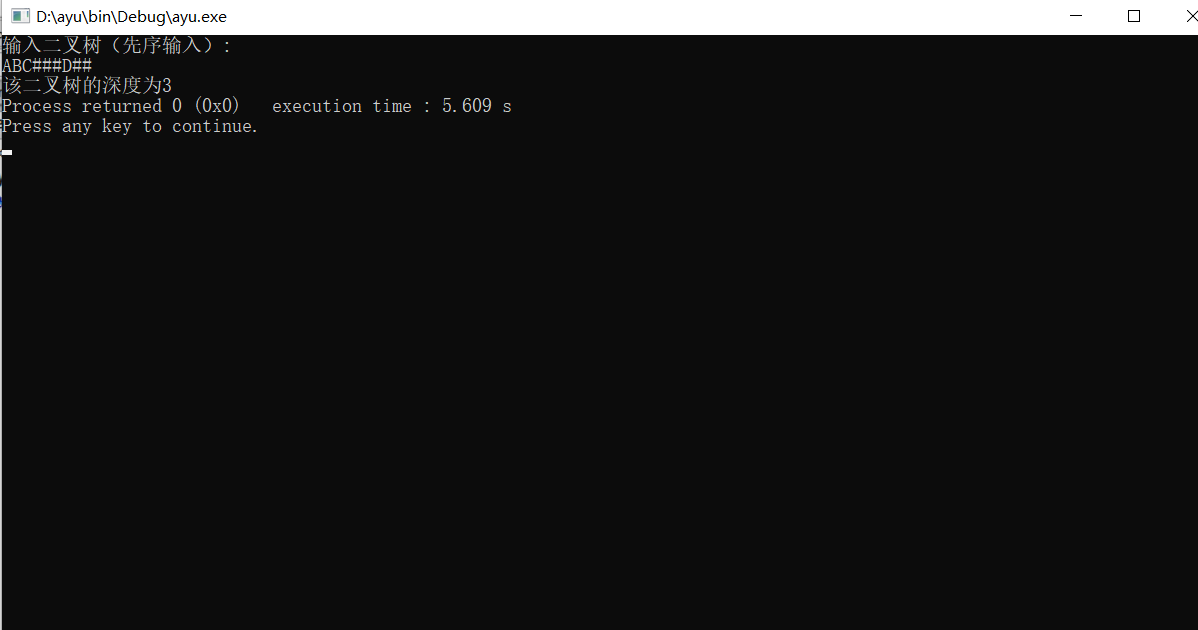
rd=deep(bt->rchild)+1; //每递归一次，rd+1

}

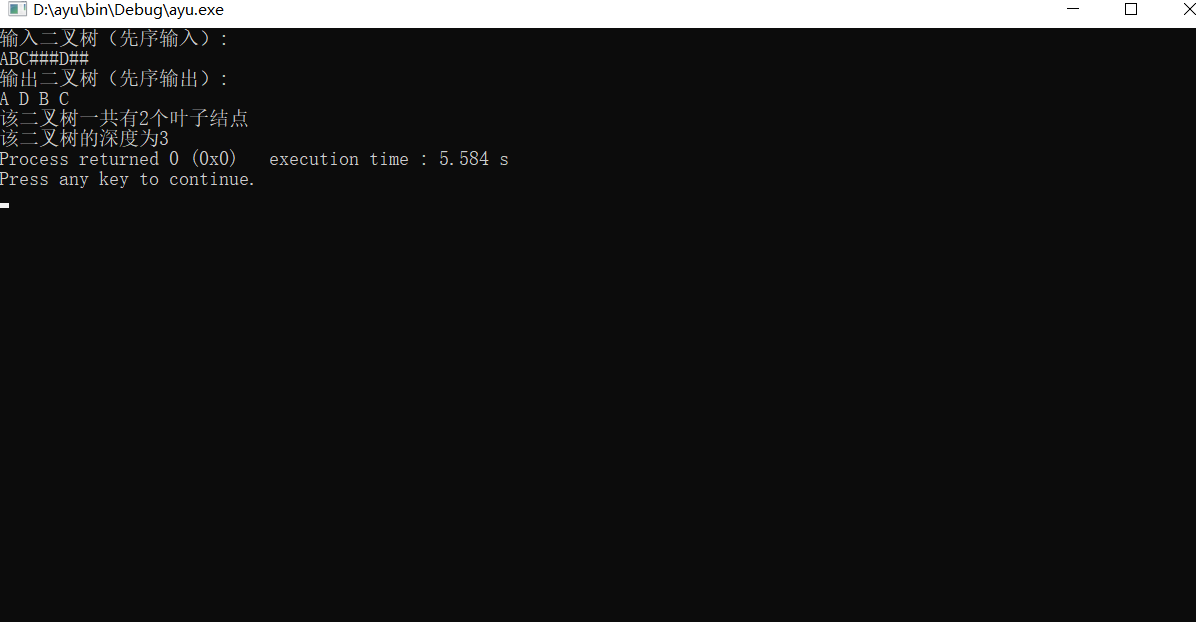
return ld>=rd?ld:rd; //返还ld与rd中最大值

}

程序运行截图



源代码



#include<stdio.h>

#include<stdlib.h>

typedef struct tree

{

char ch;

struct tree \*lchild;

struct tree \*rchild;

}Tree,\*BitTree;

Tree \*InitTree() //创建二叉树

{

Tree \*bt;

char str;

scanf("%c",&str);

if(str=='#') //若输入字符是# 则代表该出为空

return NULL;

else

{

bt=(BitTree)malloc(sizeof(Tree));

bt->ch=str; //赋值

bt->lchild=InitTree(); //递归创建左孩子

bt->rchild=InitTree(); //递归创建有孩子

return bt;

}

}

void Exchange(Tree \*bt) //交换左右子树

{

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

;

else

{

Tree \*temp=bt->lchild; //用temp暂时存储左孩子数值

bt->lchild=bt->rchild;

bt->rchild=temp;

}

if(bt->lchild) //分别判断左右孩子是否为空

Exchange(bt->lchild); //递归继续寻找可交换节点

if(bt->rchild)

Exchange(bt->rchild);//递归继续寻找可交换节点

}

void PreOrder(Tree \*bt) //先序输出

{

if(bt!=NULL)

{

printf("%c ",bt->ch);

PreOrder(bt->lchild);

PreOrder(bt->rchild);

}

}

int Leaf(Tree \*bt,int count)

{

if(bt)

{

if(bt->lchild==NULL && bt->rchild==NULL) //叶子结点左右孩子为空

{

count++;

}

count=Leaf(bt->lchild,count); //递归判断

count=Leaf(bt->rchild,count); //递归判断

}

return count;

}

int deep(Tree \*bt)

{

int ld=0,rd=0; //ld为左侧深度 rd为右侧

if(bt)

{

ld=deep(bt->lchild)+1; //每递归一次，ld+1

rd=deep(bt->rchild)+1; //每递归一次，rd+1

}

return ld>=rd?ld:rd; //返还ld与rd中最大值

}

int main()

{

Tree \*bt;

int num\_leaf=0;

printf("输入二叉树（先序输入）:\n");

bt=InitTree();

Exchange(bt);

printf("输出二叉树（先序输出）:\n");

PreOrder(bt);

printf("\n");

printf("该二叉树一共有%d个叶子结点",num\_leaf=Leaf(bt,num\_leaf));

printf("\n");

int num\_deep;

printf("该二叉树的深度为%d",num\_deep=deep(bt));

return 0;

}