**一、【实验标题】**

**实验3 二叉树的遍历**

**二、【实验目的】**

1. 了解二叉树的前序、中序、后序和层次序列排列；
2. 将C语言同二叉树的数据结构联系起来；
3. 掌握生成的二叉树的链表结构；
4. 掌握如何按层次输出二叉树的所有结点；
5. 掌握如何将动态二叉树转换为静态二叉链表。

**三、【实验环境】**

硬件: Intel®Core™i5-6500 CPU3.20GHz,8GB内存

软件: Windows 7 64位,Microsoft Visual Studio 2010

**四、【实验内容】**

创建一个二叉树，对这棵动态二叉树进行分析，将其用静态二叉链表表示。二叉树的动态二叉链表结构中的每个结点有三个字段：data，lchild（左孩子），rchild（右孩子）。静态二叉链表是用数组作为存储空间，每个数组元素存储二叉树的一个结点，也有三个字段：data，lchild，rchild。lchild和rdhild分别用于存储左右孩子的下标。

**五、【实验程序】**

#include<stdio.h>

#include<stdlib.h>

#define MAX 7

using namespace std;

typedef struct node///结点结构

{

char data;///数据域

struct node \*lchild;///左孩子指针

struct node \*rchild;///右孩子指针

}BiTNode,\*BiTree;///树中结点类型

typedef struct BiTNode\_s///树结构

{

char data;

int lchild;///左孩子下标

int rchild;///右孩子下标

}BiTNode\_s,\*BiTree\_s;

int i=0;

///前序法构建二叉树

void PreCreatBiTree(BiTree&T)

{

///按先序次序输入二叉树中结点的值（一个字符），创建二叉链表表示的二叉树T

char ch;///结点的值

if((ch=getchar())=='\*')///用\*表示空

T=NULL;///递归结束，建空树

else

{

T=(BiTNode \*)malloc(sizeof(BiTNode));///开辟新的空间

T->data=ch;///生成根结点赋值

PreCreatBiTree(T->lchild);///递归创建左子树

PreCreatBiTree(T->rchild);///递归创建右子树

}

}

///前序遍历，中左右

void PreTravel(BiTree &T)

{

if(T)

{

printf("%c ",T->data);///中

PreTravel(T->lchild);///先左子树

PreTravel(T->rchild);///后右子树

}

}

///中序遍历，左中右

void MidTravel(BiTree&T)

{

if(T)

{

MidTravel(T->lchild);

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

MidTravel(T->rchild);

}

}

///后序遍历，左右中

void AftTravel(BiTree &T)

{

if(T)

{

AftTravel(T->lchild);

AftTravel(T->rchild);

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

}

}

void LevelTravel(BiTree &T){

BiTree Queue[MAX],b;

///用一维数组表示队列，front和rear分别表示队首和队尾指针

int front,rear;

front=rear=0;

if(T) ///若树非空

{

Queue[rear++]=T;///根结点入队列

while(front!=rear) ///当队列非空

{

b=Queue[front++];///队首元素出队列，并访问这个结点

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

if(b->lchild!=NULL)

{

Queue[rear++]=b->lchild;

///左子树非空，则入队列

}

if(b->rchild!=NULL)

{

Queue[rear++]=b->rchild;

///右子树非空，则入队列

}

}

}

}

///将动态二叉树转化为用静态二叉链表

void TreeToArray(BiTree &T,BiTNode\_s a[])

{

int k;

a[i].data=T->data;

k=i;///k用于记录递归调用的每一层此结点对应的数组下标,i在外面就被初始化为0

if(T->lchild)

{

i++;

a[k].lchild=i;///将左孩子结点对应的下标赋给双亲结点

TreeToArray(T->lchild,a);///递归调用Tree\_to\_Array函数

}

else

a[k].lchild=-1;///没有左孩子结点

if(T->rchild)

{

i++;

a[k].rchild=i;///将右孩子结点对应的下标赋给双亲结点的rcjild

TreeToArray(T->rchild,a);///递归调用Tree\_to\_Array函数

}

else

a[k].rchild=-1;///没有右孩子结点

}

int main()

{

int j;

BiTree T;

BiTNode\_s a[MAX];

printf("请按先序遍历顺序输入二叉树的各节点，没有的用符号\*代替：\n");

PreCreatBiTree(T);

printf("前序遍历结果：\n");

PreTravel(T);

printf("\n");

printf("中序遍历结果：\n");

MidTravel(T);

printf("\n");

printf("后序遍历结果：\n");

AftTravel(T);

printf("\n");

printf("层次遍历结果：\n");

LevelTravel(T);

printf("\n");

TreeToArray(T,a);

printf("转化成静态二叉链表为：\n");

printf("结点编号 结点字母 左孩子 右孩子\n");

for(j=0;j<MAX;j++){

printf("%5d %10c",j,a[j].data);

if(a[j].lchild==-1) printf(" %8c",'\*');

else printf(" %8d",a[j].lchild);

if(a[j].rchild==-1) printf(" %8c",'\*');

else printf(" %8d",a[j].rchild);

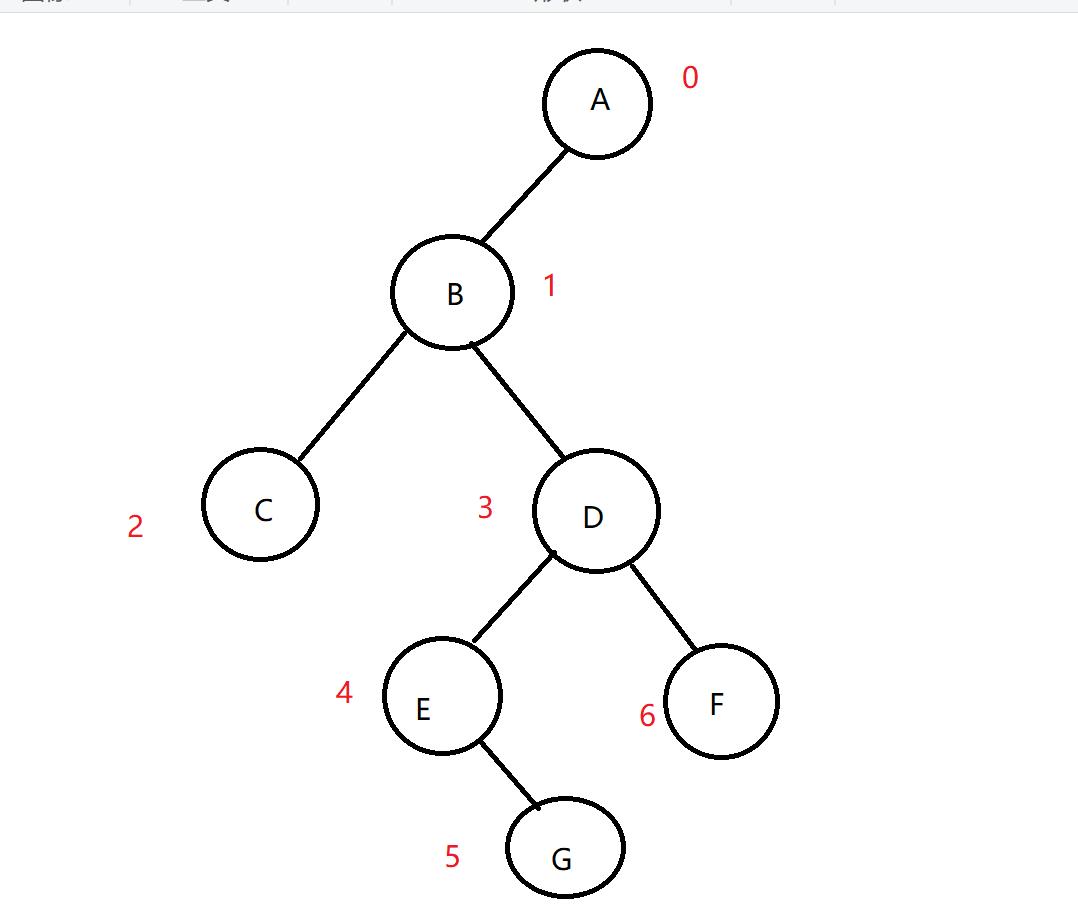
printf("\n");

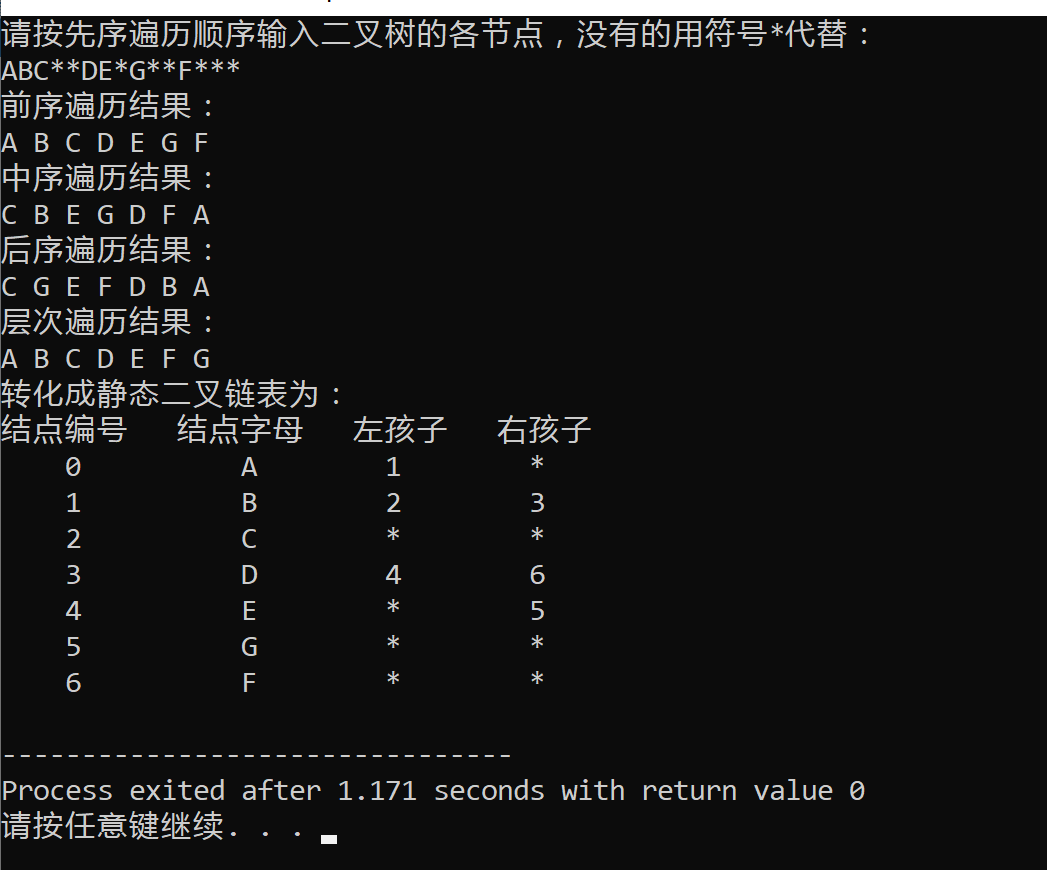
}

return 0;

}

**六、【实验运行结果】**





**七、【实验小结】**

**本次实验我学到了二叉树的前序、中序、后序和层次序列排列，前、中、后序是根据遍历根的顺序来划分的，层次序列是一层一层从上往下进行遍历。**

**通过递归来实现对二叉树的遍历。同时我还学会了如何使用链表构建二叉树，通过链表的指针来遍历结点。掌握了如何将动态二叉树转换为静态二叉链表，利用数组储存对应的结点，然后用下标直接遍历结点。**

**通过本次实验的学习，我对于二叉树的构建，遍历有了更深的理解，同时更熟练地掌握了链表和递归的使用，对于递归的先后顺序理解的更加透彻。学到了关于二叉树的许多特点和性质。**