## 实验4 二叉树的建立、遍历及应用

1. 实验目的

（1）能实现二叉树链式存储结构下的创建及遍历算法；

（2）能根据实际问题，设计二叉树存储结构并设计相应算法。

2. 实验内容

（1）第一关：二叉树链式存储结构设计及二叉树创建和遍历算法；

（2）第二关：实现二叉树中树叶结点计数

（3）第三关：实现二叉树左右子树互换算法；

详见educoder

### 第一关：

#### 任务描述

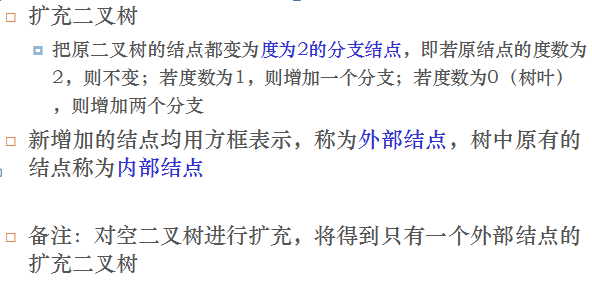
本关任务：完成二叉树的创建及遍历输出。

#### 相关知识

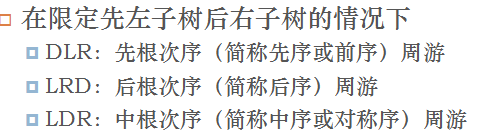
为了完成本关任务，你需要掌握：

1. 如何创建一棵二叉树；
2. 如何遍历二叉树并输出。

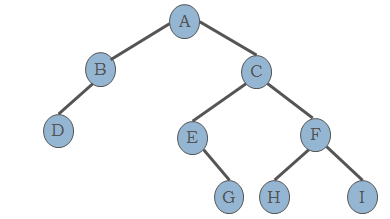
##### 扩充二叉树



##### 遍历二叉树（二叉树的周游）



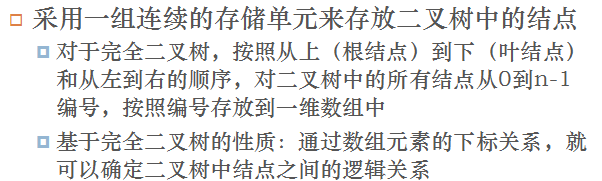
假设有二叉树如下图所示：



则先根遍历得到的结点序列是： ABDCEGFHI； 中根遍历得到的结点序列是： DBAEGCHFI； 后根遍历得到的结点序列是： DBGEHIFCA。

##### 如何存储一棵二叉树

1. 顺序存储法



2.链式存储法 除了存储结点本身的数据外，再设置两个指针字段：lchild 和 rchild，分别存放结点的左孩子结点和右孩子结点的位置。 数据结构可定义如下:

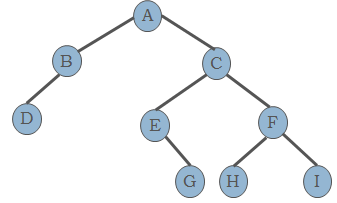
1. typedef struct node
2. {
3. char data; //存储结点的数据
4. struct node \*lchild,\*lchild ;
5. }BiTreeNode, \*BiTree;

#### 编程要求

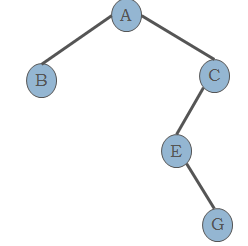
根据提示，在右侧编辑器Begin-End处补充代码，完成创建一棵二叉树，并输出该二叉树的先根、中根、后根遍历结果。

#### 测试说明

平台会对你编写的代码进行测试：



测试输入：ABD###CE#G##FH##I##； 预期输出： A B D C E G F H I D B A E G C H F J D B G E H I F C A



测试输入：AB##CE#G###； 预期输出： A B C E G B A E G C B G E C A

#include <iostream>

using namespace std;

//此处如有需要则可以自行增加自己所需要的头文件

/\*-------begin--------\*/

/\*-------end--------\*/

typedef char DataType;

//二叉树结点定义

struct node

{

   DataType data; //存放结点数据

   struct node \*lchild, \*rchild ; //左右孩子指针

};

typedef struct  node  BiTree;

typedef struct  node  \*ptree;

//函数可直接使用，功能：输出结点数据

void print(DataType d)

 {

   cout<<d<<" ";

 }

/\*

函数名：createBiTree

函数功能：创建二叉树，要求输入二叉树的先根序序列(具体输入方式请看左侧说明)，并创建对应二叉树，并返回二叉树的根结点指针

参数：无

返回值：二叉树的根结点指针

\*/

BiTree \*createBiTree() {

//请在此处填写代码，完成创建二叉树并返回二叉树根结点指针的功能

/\*-------begin--------\*/

/\*-------end--------\*/

}

/\*

函数名：preOrder

函数功能：先根遍历二叉树

参数：二叉树根结点指针

返回值：无

\*/

void preOrder(BiTree \*T)

{

//请在此处填写代码，完成先根遍历二叉树功能

/\*-------begin--------\*/

/\*-------end--------\*/

}

/\*

函数名： inOrder

函数功能：中根遍历二叉树

参数：二叉树根结点指针

返回值：无

\*/

void inOrder(BiTree \*T)

{

    //请在此处填写代码，完成中根遍历二叉树功能

/\*-------begin--------\*/

/\*-------end--------\*/

}

/\*

函数名：postOrder

函数功能：后根遍历二叉树

参数：二叉树根结点指针

返回值：无

\*/

 void postOrder(BiTree \*T)

    {

      //请在此处填写代码，完成后根遍历二叉树功能

/\*-------begin--------\*/

/\*-------end--------\*/

    }

int main(void)

{

    BiTree    \*T;

     T =  createBiTree(   ); //调用创建二叉树功能，得到二叉树的根结点指针

    preOrder(  T );//调用先根遍历二叉树，按先根遍历顺序输出二叉树结点功能

    cout<<endl; //换行

    inOrder(T);//调用中根遍历二叉树，按中根遍历顺序输出二叉树结点功能

    cout<<endl;

    postOrder(T);//调用后根遍历二叉树，按后根遍历顺序输出二叉树结点功能

   return 0;

}

### 第二关：

#### 任务描述

统计二叉树的叶子结点个数。 （1）输入二叉树的先根序序列，构造二叉树 （2）输出所构造的二叉树的叶子个数

#### 相关知识（略）

#### 编程要求

根据提示，在右侧编辑器Begin-End处补充代码，完成本关任务。

#### 测试说明

平台会对你编写的代码进行测试：

测试输入（注意＃是特殊的字符，并不会被插入二叉树中）：ABC##D##E#F#G## 预期输出：3

测试输入：eba###f## 预期输出：2

//请自行在下面自行完整实现代码，用于计算二叉树叶子结点个数

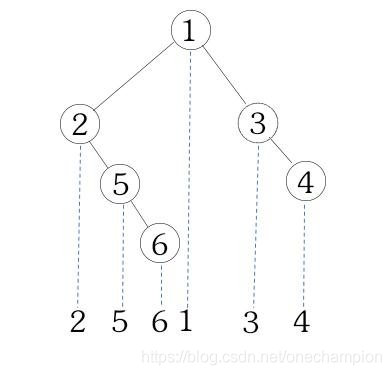
/\*-------begin--------\*/

/\*-------end--------\*/

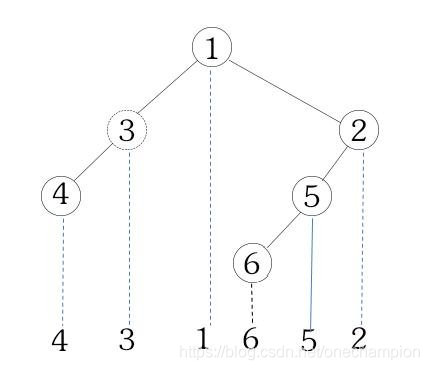
**方法：递归访问二叉树，统计左右子树都为NULL的结点个数。**

### 第三关：

#### 任务描述

本关任务：二叉树左右子树互换。 原二叉树 

交换后的二叉树



#### 提示

可以使用递归完成，递归需要明白递归结束条件 如果结点不为NULL，则交换左右子树（用递归即可）

#### 编程要求

根据提示，在右侧编辑器完整实现要求。

#### 测试说明

平台会对你编写的代码进行测试：

测试输入（树的先根序，具体二叉树见上）：12#5#6##3#4## 预期输出(转换后二叉树的先根序遍历序列)：124256

//请自行在下面自行完整实现代码，实现二叉树左右子树交换

/\*-------begin--------\*/

/\*-------end--------\*/

参考讲解：

1. typedef struct BiTNode {
2. char data;
3. struct BiTNode \*lchild,\*rchild;
4. } BiTNode,\*BiTree;
5. void exchange(BiTree t)
6. {
7. BiTree p;
8. if (t!=NULL)
9. {
10. p=t->lchild;
11. t->lchild=t->rchild;
12. t->rchild=p;
13. exchange(t->lchild);
14. exchange(t->rchild);
15. }
16. }