我们对图 8.6 所示二叉树分别施用前面的遍历算法，结果如下：

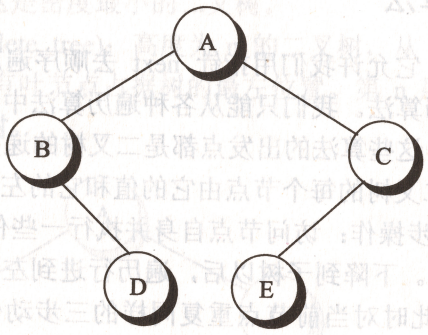

图 8.6　二叉树

前序遍历(NLR)：ABDCE
中序遍历(LNR)：BDAEC
后序遍历(LRN)：DBECA
层次遍历：ABCDE

8.1.5　二叉树遍历算法的应用

1．应用：叶节点计数

叶结点的指针域为 NULL。考查叶节点的个数，必然涉及节点的遍历，并检查
点的指针域是否为空，而遍历的顺序则无关紧要，所以，我们可以使用任意一
算法。

2．应用：计算树深度

设根节点左子树的深度为 h_L，右子树的深度为 h_R，显然，二叉树的深度 h 应为

$$h=1+\max\{h_L, h_R\}$$

这样，为求得树的深度，我们必须知道其左右子树的深度，递归地应用这个算
计算左子树的深度，再计算右子树的深度，然后计算结点所处层次，逐层递归。

显然，计算树的深度需要采用后序遍历算法，先访问子节点(左、右子树)，再
节点。

至于递归遍历的结束条件，我们定义空树的深度为-1，这样，我们就使得仅有
点的子树深度为 0。

图 8.7 显示了后序遍历对节点的访问顺序。每个节点的下脚标显示了以该节点为
子树的深度，节点旁的标签显示了遍历访问顺序。如 B_2、#4 就说明节点 B 的子树
为 2，访问顺序为 4(之前依次访问了节点 G、D、E)。

3．应用：树的复制与删除

复制与删除整棵树的应用函数将引入新的概念，并为开发树类所必需的复制构
和析构函数作好准备。函数 copyTree()以原始树为参数建立其复制品，deleteTree()