**一：二叉树**

1. 定义：一棵二叉树是结点的一个有限集合，该集合或者为空，或者是由一个根节点加上两棵别称为左子树和右子树的二叉树组成。

2.二叉树的特点：  
 a.每个结点最多有两棵子树，即二叉树不存在度大于2的结点。  
 b.二叉树的子树有左右之分，其子树的次序不能颠倒。

3.二叉树的基本性质

(1). 在二叉树的第ｉ（ｉ>=１）层最多有２＾(ｉ - １)个结点。  
(2). 深度为k(k>=0)的二叉树最少有k个结点，最多有２＾ｋ－１个结点。  
(3). 对于任一棵非空二叉树，若其叶结点数为n0，度为2的非叶结点数为n2，则ｎ0 = ｎ2 ＋１。  
(4). 具有n个结点的完全二叉树的深度为int\_UP（log(2，ｎ+1)）。  
(5). 如果将一棵有n个结点的完全二叉树自顶向下，同一层自左向右连续给结点编号１，２，３，．．．．．．，ｎ，然后按此结点编号将树中各结点顺序的存放于一个一维数组，并简称编号为i的结点为结点i（ ｉ>=１ && ｉ<=ｎ）,则有以下关系：  
a.若 ｉ= 1，则结点i为根，无父结点；若 ｉ> 1，则结点 i 的父结点为结点int\_DOWN（ｉ / ２）;  
b.若 ２＊ｉ <= ｎ，则结点 ｉ 的左子女为结点 ２＊ｉ；  
c.若２＊ｉ＜＝ｎ，则结点ｉ的右子女为结点２＊ｉ＋１；  
d.若结点编号ｉ为奇数，且ｉ！＝１，它处于右兄弟位置，则它的左兄弟为结点ｉ－１；  
e.若结点编号ｉ为偶数，且ｉ！＝ｎ，它处于左兄弟位置，则它的右兄弟为结点ｉ＋１；

1. 二叉树的存储方式  
   二叉树一般可以使用两种结构存储，一种顺序结构，一种链式结构。

顺序存储表示的描述如下：

#define MAX //二叉树的最大节点数

typedef elemtype SqBitree[MAX] //0号元素存放根结点

SqBiTree bt;

链式存储结构表示的描述如下：

用链表来表示一颗二叉树，即用链指针来指示元素的逻辑关系。

链表中的每一个结点有三个域组成，除了数据域外，还有两个指针域，分别用来给出该结点的左孩子和右孩子所在的链结点的存储地址。

typedef struct node

{ elemtype data;

struct node \*lchild,\*rchild;

} BiTNode,\*BiTree;

将BiTree定义为指向二叉链表结点结构的指针类型.

1. 二叉树的遍历

（1）先序遍历

根节点——>左子树——>右子树

Void PreOrder(Bitree bt)

{

//先序遍历二叉树bt

If(bt==NULL) return;

Visite(bt->data);

PreOrder(bt->lchild);

PreOrder(bt->rchild);

}

1. 中序遍历

左子树——>根节点——>右子树

Void InOrder(Bitree bt)

{

//先序遍历二叉树bt

If(bt==NULL) return;

InOrder(bt->lchild);

Visite(bt->data);

InOrder(bt->rchild);

}

1. 后序遍历

根节点——>左子树——>右子树

Void PostOrder(Bitree bt)

{

//先序遍历二叉树bt

If(bt==NULL) return;

PostOrder(bt->lchild);

PostOrder(bt->rchild);

Visite(bt->data);

}