约定：层序编号：二叉树从根结点开始按照从上到下，从左到右的顺序从1开始给结点编号。

二叉树(binary tree)

定义：

一个有限的结点集合。这个集合或者为空，或者由一个根结点和两棵互不相交的称为左子树（subtree）和右子树的二叉树组成。

二叉树是一种特殊的树形数据结构，二叉树的定义也是递归的，这决定了对二叉树的算法设计往往设计递归思想

二叉树与二次树的区别：

二次树必须有一个度为2的结点，而二叉树无要求；

二叉树严格区分左右子树，而二次树无要求。

**二叉树的其中两种形态**

满（full）二叉树：

所有分支结点都有左右子树，并且叶子结点都集中在最高层。/ 高度为h且有2h-1个结点的二叉树。

特点：

所有叶子结点都在最高层；

只有度为0和2的结点。

完全（complete）二叉树：

满足: 1) 最下面两层的结点的度小于等于2；

2) 最下面一层的叶子结点靠左排列 的二叉树。

特点：

所有叶子结点只出现在最下面两层；

最下面一层的叶子结点依次排列在最左边；

如果有度为1的结点，只可能有一个，并且这个结点只有左孩子；

按层序编号时，一旦出现编号为i的结点是叶子结点或者是只有左孩子的结点，则编号大于i的结点都为叶子结点；

当结点总数是奇数时，n1=0，偶数时，n1=1。

两种形态的关系：满二叉树是完全二叉树的特例

**二叉树的性质**

1. 二叉树的叶子结点数等于双分支结点数加1

证明：

由树的性质得：

n = n0 + n1 + n2（结点总数） = n1 + 2n2 + 1（所有结点度数之和加1 ）

推出：n0 = n2 + 1

1. 同树的性质二（m次树第i层的最大结点数）
2. 同树的性质三（由树高h确定树的最大结点数）
3. 完全二叉树中**层序编号**为i的结点(1<= i <=n , n>=1，n为结点数)

有以下性质：

1. 若i <= (n/2)向下取整，则编号为i的结点为分支结点，否则为叶子结点；

*性质1)等价于：完全二叉树中最后一个叶子结点的层序编号n除以2向下取整，将得到最后一个分支结点的层序编号；或者这样理解：某结点的双亲结点的层序编号可由其编号除以2向下取整得到。*

1. 若n为奇数，则每个分支结点都有左右孩子，若为偶数，则最大编号分支结点只有左孩子；

*奇数的情况：一个根结点 + 偶数个孩子结点（后继结点）*

*偶数的情况：一个根结点 + 偶数个右孩子结点 + 奇数个左孩子结点*

1. 编号为i的结点的左孩子结点编号为2\*i，右孩子编号为2\*i+1；
2. 编号为i的结点的双亲结点编号为(i/2)向下取整
3. 具有n个（n>0）结点的完全二叉树的高度为（log2(n+1)）向下取整，或（log2n）向上取整加1

二叉树的存储结构

**顺序存储结构**

存储结构实现：一维数组，

结点类型声明实现代码：

typedef char ElemType；

typedef ElemType SqBinTree[MaxSize];

为了统一二叉树的层序编号，在C/C++中，数组首元素位置不用。

最适情况：完全二叉树、满二叉树

不太适用：一般二叉树

因为既要存储结点值又要存储结点之间的关系，因此不能直接按顺序将结点存储到数组中，而要采用方法：先用空结点补全为一棵完全二叉树，然后进行层序编号，再将非空结点存储到对应编号的数组位置上。这会导致的问题是：有些二叉树的形态会浪费较大或极大的存储空间。如一棵树高为h的右单支树，实际结点有h个，但需要2h-1个结点空间存储。

优点：数组下标对应逻辑结构的层序编号，查找某类结点可直接计算得出，如编号为i的结点的孩子结点（2\*i和2\*i+1）及双亲结（i/2向下取整）、确定编号为i的结点的所在层数等等。

缺点：顺序存储结构的固有缺陷：不便于插入和删除操作。

总结：顺序存储结构比较适合存储完全二叉树、满二叉树。

**二叉链**（binary link list）

存储结构：

结构体

实现代码：

结点默认按层序编号的排序，结点之间的逻辑关系用指针域实现。

优点：相对于顺序存储结构，二叉链的空间随情况而分配，具有很好的灵活性及很好的存储空间利用率，因此适用于二叉树的所有形态。

缺点：链式存储结构的不足：不具备随机存取特性，但因为树这种数据结构是递归的，所以大多数情况下可以采用递归算法实现对数据结点的访问。

**二叉树ADT**

{

数据对象：

D = { ai  | 1<=i<=n，n>=0 }

数据关系：

R = { <ai , aj> | 1<= I,j <=n }

基本运算：

CreateBTree(&b);

DestroyBTree(&b);

DispBTree(b);

BTreeHeight(b);

LChild(t,p);

RChild(t,p);

Parent(t,p);

……

}

**二叉树的基本运算**

最基本运算：

二叉树的遍历：前序遍历（访问顺序：根、左子树、右子树）、中序遍历、后序遍历

二叉树的遍历算法通常采用递归的方法实现，因为树这种数据结构是递归的。当然，也有非递归的遍历算法，下面对以递归遍历为基础的算法进行总结。