Tree是一个从起始点开始的树形分散struct。

再来稳固一下struct

struct node

{

int value;

string name; //这三个是node内包含的内容

double salary;

node \*next，\*pre；// 这是这个node的两个pointer，分别指向前一个node和后一个node。

}

在main内调用node。

Int main

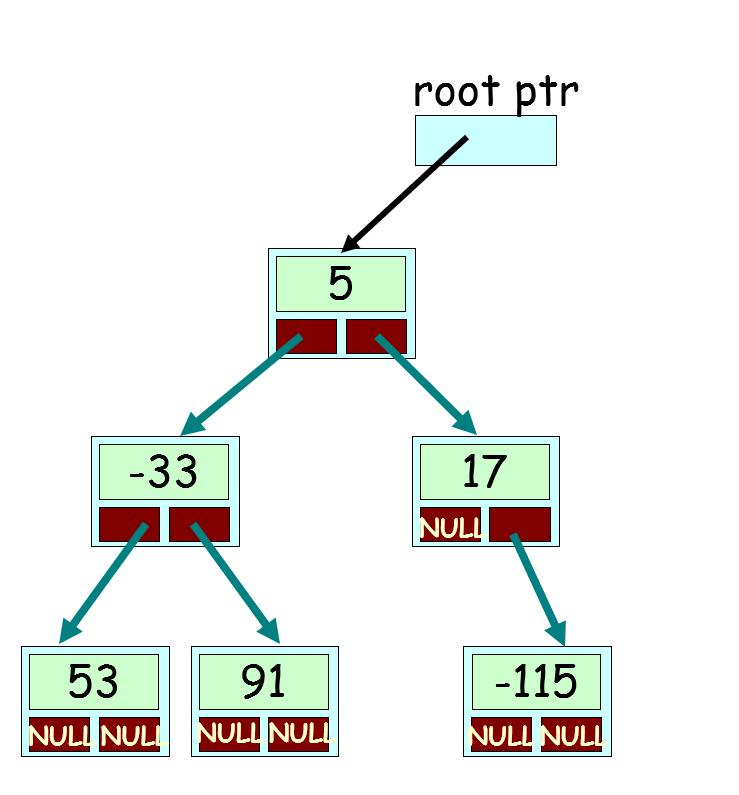
{

node example;

example -> value; //这是读取该node里的数据。

example -> next; //这是指向下一个node。

}



但在tree的模式下，我们把next 和 pre改为left和right。

从起始点出发，left为下放左边一个数据，right为下放右边一个数据。

起始点被叫做 root ptr。一般不储存任何数据，和head node类似。

第一个数据点称为root node。

Root node下有再分支的node称为child

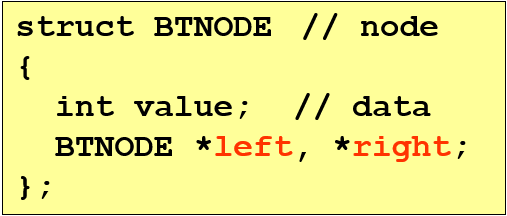
没有再分支的node称为leaf node.

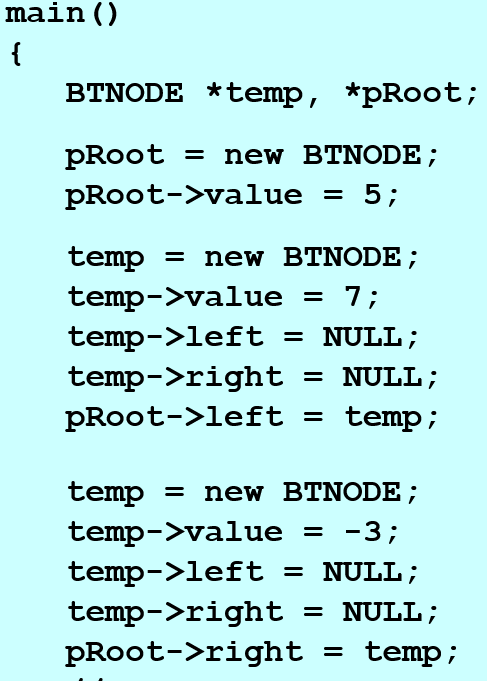
如果需要从起始点分出更多的分支，就可以在node ptr设置时设置多个节点。

Node \*ptr1; \*ptr2; \*ptr3; \*ptr4; \*ptr5;…….

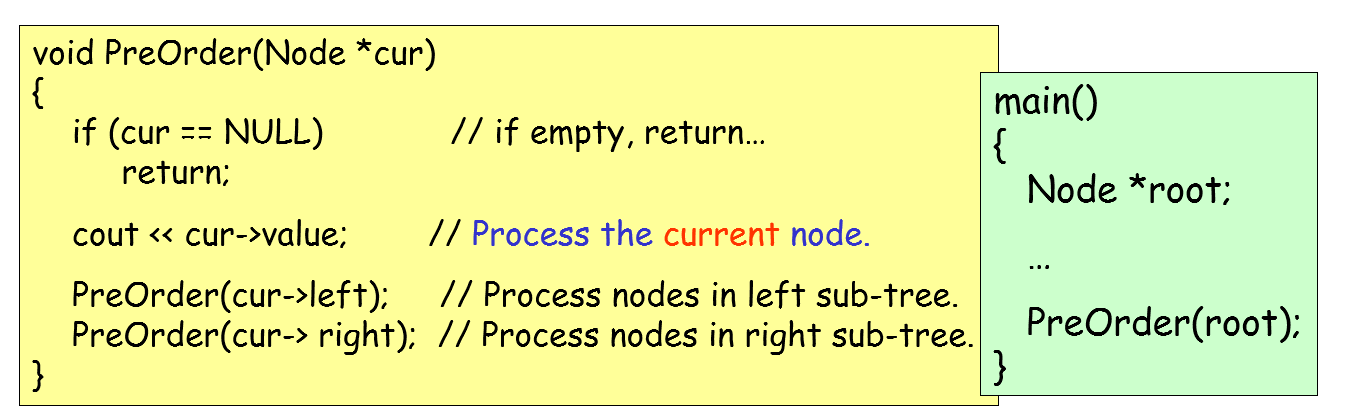
或者Node \*ptr[xx]；

**Simple tree example：**



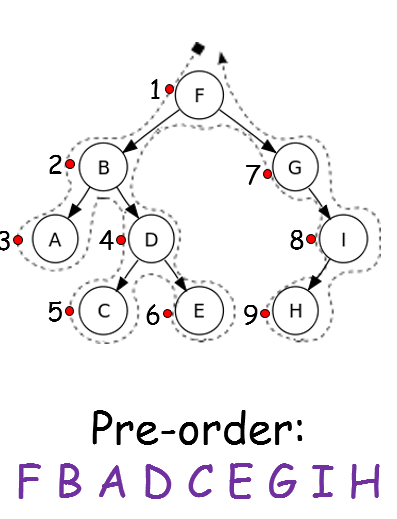


Pre-Order浏览法：其中带有自循环function，左右两个分别会最终进入null的node。所以leaf node相当于tree的终结。

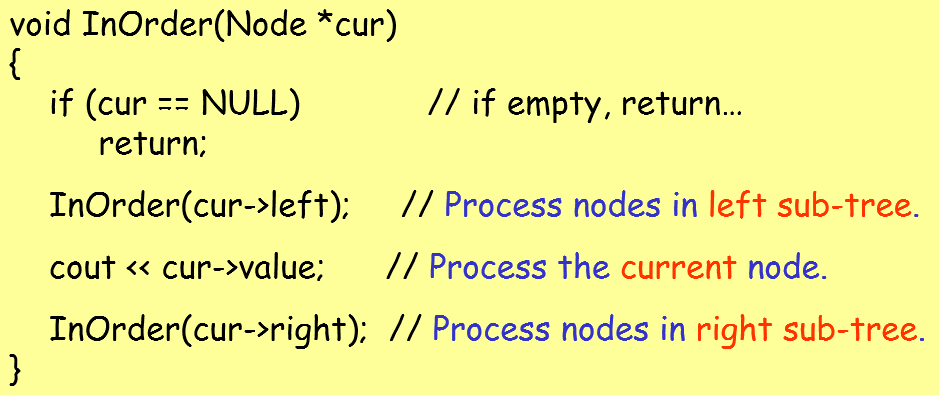


Full binary Trees 是指所有leaf node都有同样的深度 并且 每个分支都有2个children.

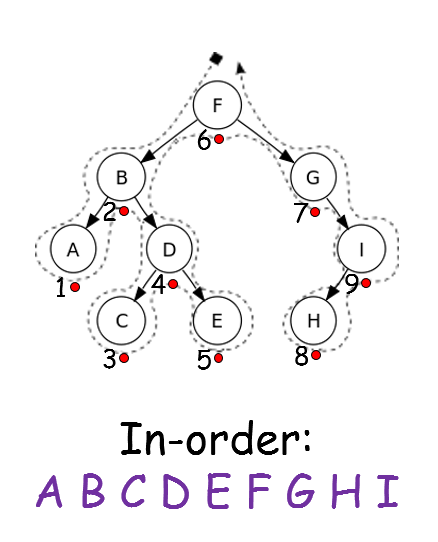
Pre-Order 从图形上看是从起始点开始沿着左边画一条顺时针的线



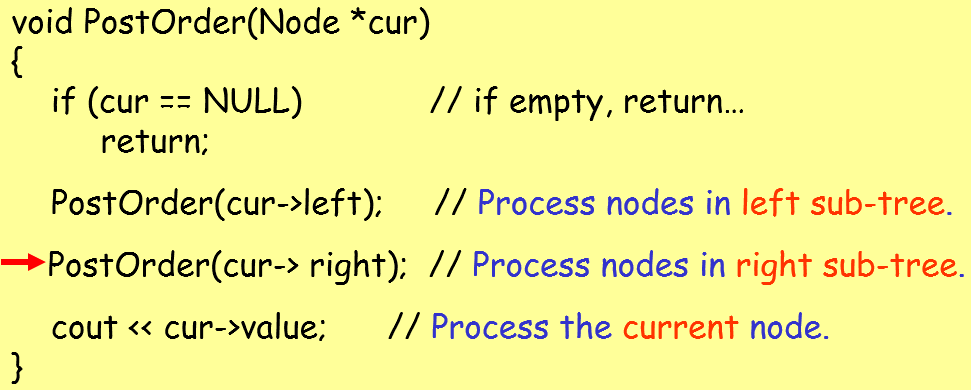
In-Order 浏览法：



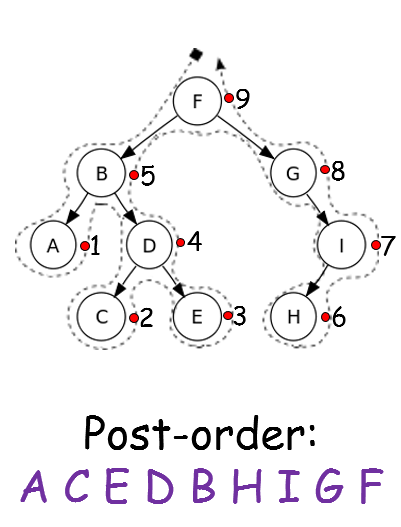
In-order是从最左边最深的一个leaf开始往上一阶，然后往右过度。



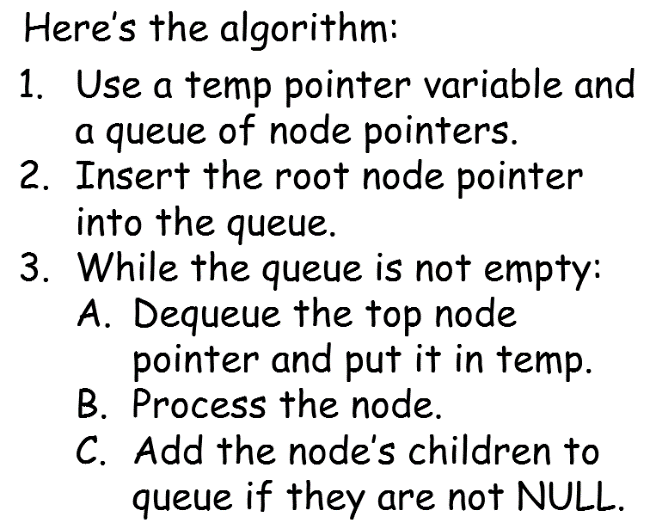
Post-Order 浏览法：

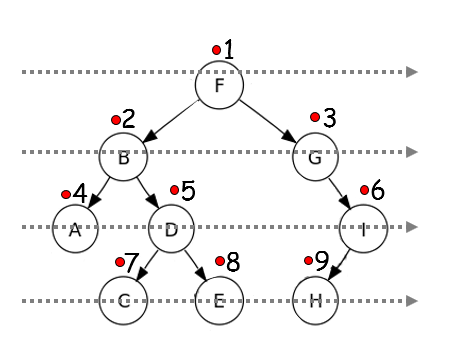


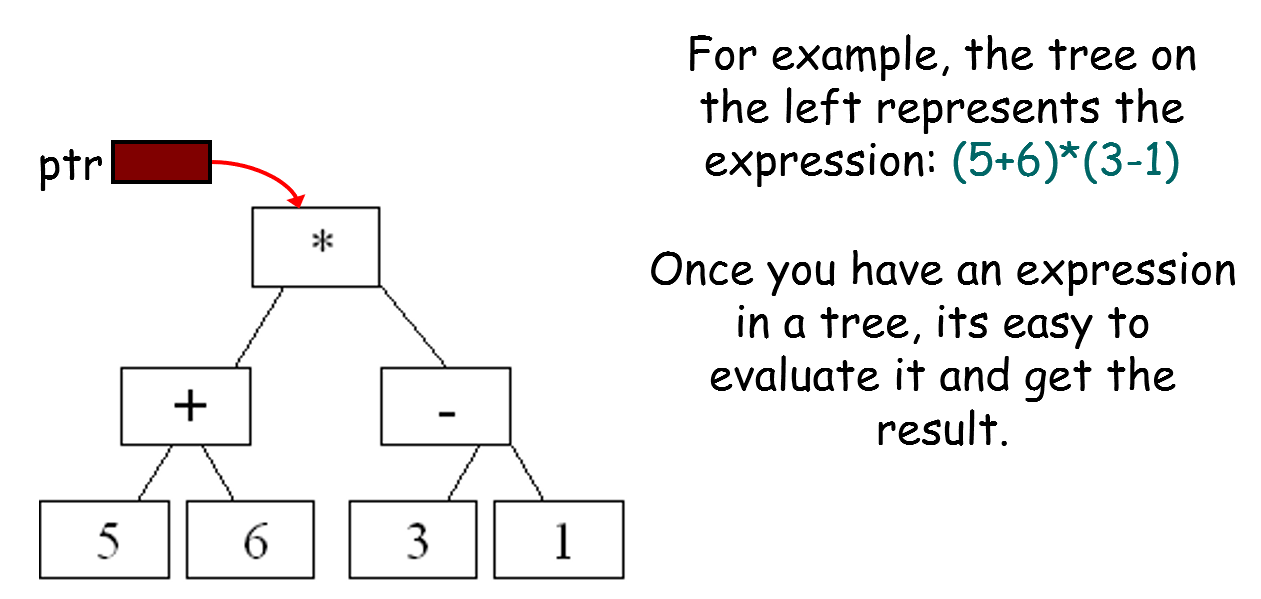
Post-Order是从最左开始往右最深处延伸

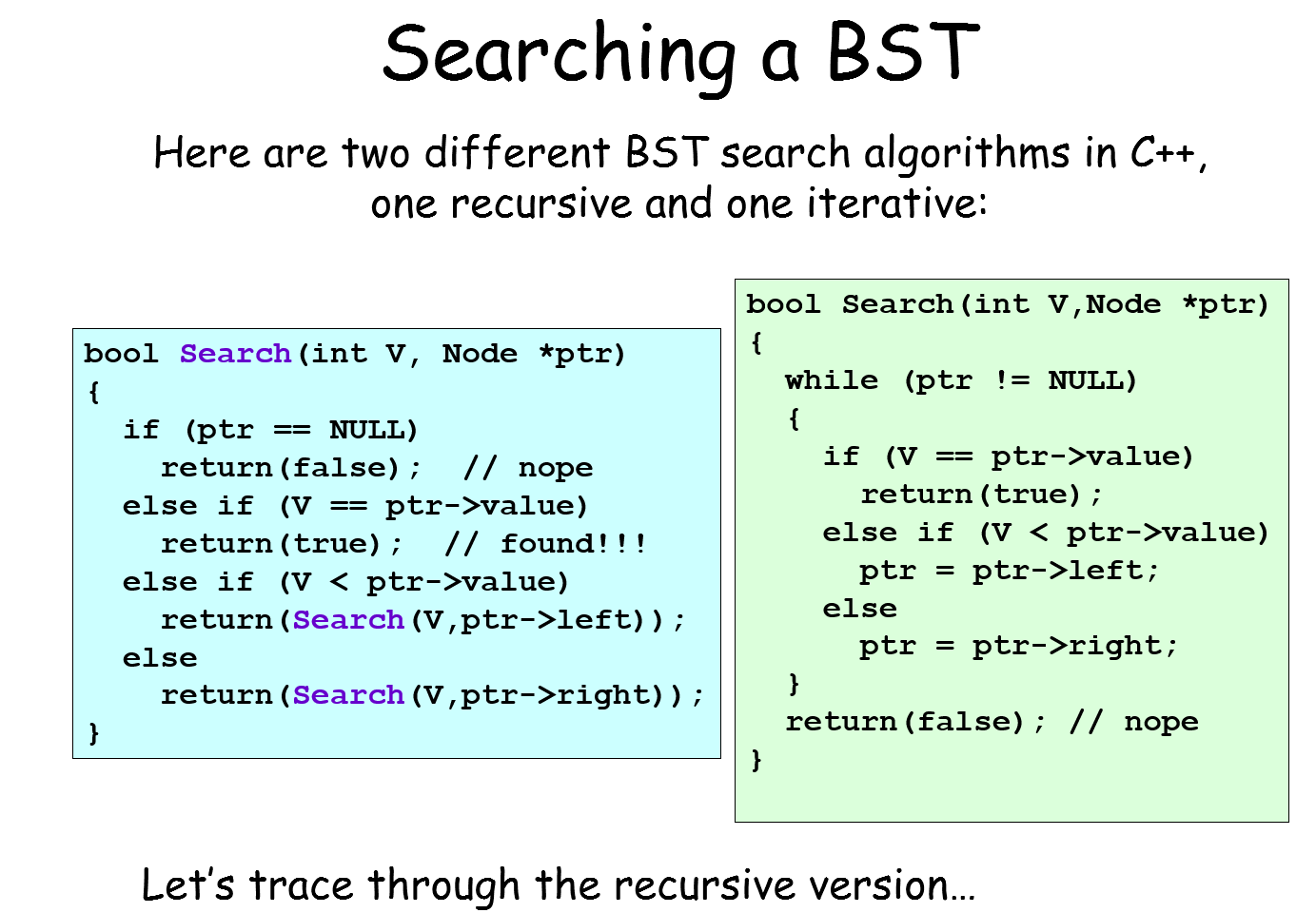


**Level order Traversal：是指在每一阶node从左到右浏览，然后再到下一个level。**





这几种方式就是计算器通过不同的运算符号，以此来运行不同的计算方式  




PPT中54页后很重要