前中后序建立树或者直接历遍

代码实现

```
void postOrder(int root,int start,int end)
{
    if (start > end)
        return;
    int index = start;
    while (inOrder[index] != preOrder[root] )
        index++;
    postOrder(root + 1, start, index - 1);
    postOrder(root + index - start + 1, index + 1, end);
    cout << preOrder[root];
    return;
}
```

简单介绍：

这时利用前序中序直接输出后序的函数。直接把输出语句改成将数据赋值到节点，就变成一个建树函数。

代码拓展：后序中序输出前序函数

```
void pre_order(int root,int start,int end)
{
    if (start > end)
        return;
    int index = start;
    while (inOrder[index] != postOrder[root])
        index++;
    cout << postOrder[root];
    pre_order(root - end + index - 1, start, index - 1);
    pre_order(root - 1, index + 1, end);
    return;
}
```

简单介绍：

作用相同，改成了后序中序出前序。

思路：

1. 参数意义：root为前序（后序）中根节点的位置，start和end是中序的起点下标和终点下标；
2. 递归出口：当发现end比start小，说明已经完成，可以退出了；
3. 递归主体：
    - 令index为start，自加，直到找到根节点在中序序列的位置。

- 输出或者建树操作.....
- 左子树参数：
  - 前转后

    root：root+1（前序特点）

    start：start

    end：index-1（根节点坐标前一个是左子树的end）
  - 后转前

    root：root-end+index-1（计算右子树长度为end-index，减掉之后-1得root坐标）

    start：start

    end=index-1
- 右子树参数
  - 前转后

    root：root+index-start+1（左子树长度为index-start，加上去之后+1得root坐标）

    start：index+1

    end：end
  - 后转前

    root：root-1（后序特点）

    start：index+1

    end：end
- 然后return；

4. 总结：

   将index找到之后，左子树长度为index-start，右子树长度为end-index。自己分析是是左是右，然后root+length之后再+1或者root-length之后-1；