2018/11/07　19：01

明确了一些操作的定义：

二叉树为每个节点最多有两个儿子节点（左儿子节点和右儿子节点）的树。

前序遍历：根结点 ---> 左子树 ---> 右子树。

中序遍历：左子树 ---> 根结点 ---> 右子树。

后序遍历：左子树 ---> 右子树 ---> 根结点。

节点的高：节点 ni 的高（height）为从 ni 到一片树叶的最长路径。所有的树叶（没有儿子节点的节点）的高都为 0。一棵树的高等于它的根的高。

2018/11/08　21：33

完成了框架的搭建

树的节点：

```
struct Node {
    char it;
    Node* lc;
    Node* rc;
    explicit Node(char it, Node* lc = NULL, Node* rc = NULL);
};
```

二叉树 ADT：

```
class BinTree {
private:
```

```
    Node* root;
    void removeAll(const Node* node);
public:
    BinTree();
    ~BinTree();
    void createBinTree(Node* &node);
    void visit(const Node* node);
    void preOrder(const Node* node);
    void inOrder(const Node* node);
    void postOrder(const Node* node);
    const int getHeight();
    int count(Node* node);
    Node* &getRoot();
};
```

2018/11/10　20：16

在实现建树操作的时候遇到了问题，由于输入顺序为先序遍历的结果，所以正常情况下输入与先序遍历的输出是完全相同的，但是事实是，中序遍历的输出对应了输入，经过检查发现是建立新节点的操作用在了错误的时间，错误如下：

```
30  void BinTree::createBinTree(Node* &node) {
31      char tmp;
32      std::cin >> tmp;
33      if (tmp == '/') {
34          node = nullptr;
35      }
36      else {
37          createBinTree(node->lc);
38          node = new Node(tmp);
39          createBinTree(node->rc);
40      }
41  }
```

38 行建立新节点的操作应该在 36、37 行之间执行。

2018/11/11　12：07

完成了实验三，在实现左孩子右兄弟表示法的时候有一些自己的思考，明明二叉树已经可以分辨左右了，为什么还要用这种表示法呢？左孩子右兄弟表示法是应用于这个数任意一个节点的度数不确定的情况下，每个子节点没有左右之分，实现起来更加容易，空间效率更高，并且实现方式更具灵活性。