1. 队列的顺序存储
   1. 队列符合先进先出的数据结构
   2. 对外接口
      1. 初始化队列 init
      2. 入队 push
      3. 出队 pop
      4. 返回队头 front
      5. 返回队尾 back
      6. 返回队列大小 size
      7. 判断是否为空 isEmpty
      8. 销毁队列 destroy
   3. 测试
2. 队列的链式存储
   1. 利用链表模拟出 先进先出的数据结构
   2. 对外接口与顺序存储一致
   3. 测试
3. 树的基本概念
   1. 根 没有前驱节点
   2. 叶子 没有后继的节点
   3. 双亲 直接前驱
   4. 孩子 直接后继
   5. 节点的度 直接后继的个数
   6. 树的度 节点的度中最大值
   7. 树的高度/深度 树的层数
   8. 表示法：左孩子右兄弟，将多叉树转为二叉树
4. 二叉树的基本性质
   1. 性质1 不存在度大于2的节点，最多有两个分支
   2. 性质2 二叉树第i层上 最多有2 ^(i-1) 个节点
   3. 性质3 深度为k的二叉树 最多有 2 ^ k -1 个节点
      1. 度为2 的节点 + 1 = 叶子数
   4. 满二叉树 深度为k的二叉树并且节点数量为 2 ^ k -1
   5. 完全二叉树 除了最后一层结点外，上面是一颗满二叉树，最后一层结点不可以缺少左边的节点
   6. 性质4 具有n个结点的完全二叉树的深度必为⎣log2n⎦+1
   7. 性质5对完全二叉树，若从上至下、从左至右编号，则编号为i 的结点，其左孩子编号必为2i，其右孩子编号必为2i＋1；其双亲的编号必为i/2（i＝1 时为根,除外）
5. 二叉树递归遍历
   1. 二叉树有递归特效，利用特效可以实现递归遍历
   2. 遍历分为3中
      1. 先序 先根 再左 再右
      2. 中序 先左 再根 再右
      3. 后序 先左 再右 再根
   3. 利用代码测试
6. 二叉树常规编程
   1. 求叶子数量
   2. 求树高度
   3. 拷贝二叉树
   4. 释放二叉树
7. 二叉树非递归遍历
   1. 利用栈可以实现二叉树的非递归遍历
   2. 1 、 将根节点压入栈中
   3. 2 、 执行循环
   4. 2.1 获取栈顶元素
   5. 2.2 弹出栈顶元素
   6. 2.3 如果栈顶标志位真 直接输出
   7. 2.4 如果标志位假 将标志改为真
   8. 2.5 将 右子树 左子树 根 入栈
8. 插入排序
   1. 实现升序
   2. 从i+1开始取遍历数组
   3. 如果arr[i] < arr[i-1] 满足条件 利用temp值保存住 arr[i]数据
   4. 指向内侧j循环
      * 1. int j = i - 1;
   5. for (; j >= 0 && temp < arr[j];j--)
   6. {
   7. //数据后移
   8. arr[j + 1] = arr[j];
   9. }
   10. //将j+1 的位置设置为 temp
   11. arr[j + 1] = temp;