1. 队列 Queue
   1. 符合 先进先出的数据结构
   2. 术语：
      1. 入队 push
      2. 出队 pop
      3. 返回队列大小 size
      4. 判断是否为空 isEmpty
      5. 队头元素 front
      6. 队尾元素 back
2. 队列的顺序存储
   1. 利用数组模拟出 先进先出的数据结构
   2. 接口：
      1. 初始化队列 init
      2. 入队 push
      3. 出队 pop
      4. 返回队列大小 size
      5. 判断是否为空 isEmpty
      6. 队头元素 front
      7. 队尾元素 back
      8. 销毁队列 destroy
3. 队列的链式存储
   1. 利用链表模拟出 先进先出数据结构
   2. 设计：
      1. 节点 只维护指针域
      2. 队列结构体：
         1. struct QueueNode pHeader; 头节点
         2. int m\_Size; 队列大小
         3. struct QueueNode \* pTail; 尾节点指针
   3. 接口：
      1. 初始化队列 init
      2. 入队 push
      3. 出队 pop
      4. 返回队列大小 size
      5. 判断是否为空 isEmpty
      6. 队头元素 front
      7. 队尾元素 back
      8. 销毁队列 destroy
4. 树的基本概念
   1. 根 没有前驱
   2. 叶子 没有后继
   3. 双亲 直接前驱
   4. 孩子 直接后继
   5. 节点的度 直接后继的数量
   6. 树的度 节点的度中取最大的值
   7. 树的高度/深度 最大层数
   8. 表示法：图形、广义 、左孩子右兄弟（将多叉树转为二叉树）
5. 二叉树的基本性质
   1. 性质1: 在二叉树的第i层上至多有2^(i-1)个结点（i>0）
   2. 性质2: 深度为k的二叉树至多有2^k - 1个结点（k>0）
   3. 性质3: 对于任何一棵二叉树，若度为2的结点数有n2个，则叶子数（n0）必定为n2＋1 （即n0=n2+1） 度为2 数 + 1 = 叶子数
   4. 性质4: 具有n个结点的完全二叉树的深度必为⎣log2n⎦+1
      1. (如 log2 (15)    点击 15  log  /  2  log =)
   5. 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）
   6. 满二叉树 深度为k的数 并且节点数量为 2^k - 1
   7. 完全二叉树 除了最后一层，上面是一颗满二叉树，在最后一层上只缺少右边的若干[结点](http://baike.baidu.com/view/549491.htm)。
6. 二叉树的递归遍历
   1. 先序遍历 ： 先根 再左 再右
   2. 中序遍历 ： 先左 再根 再右
   3. 后序遍历 ： 先左 再右 再根
7. 二叉树编程
   1. 求二叉树叶子数量
      1. 左子树与右子树都同时为NULL，称为叶子
   2. 求二叉树高度
      1. 左子树高度 与 右子树高度比 ，取大的值 +1 就是这个树的高度
   3. 拷贝二叉树
      1. 先拷贝左子树
      2. 再拷贝右子树
      3. 再创建根节点 挂载拷贝出的左右子树，返回给用户
   4. 释放二叉树
      1. 利用递归特性释放二叉树
8. 二叉树非递归遍历
   1. 1、将根节点 入栈
   2. 2、只要栈中元素个数大于 0 执行循环
   3. 获取栈顶元素
   4. 出栈
   5. 如果标志位真 直接输出 并且执行下一次循环
   6. 如果为假 将标志改为真
   7. 将右子树 左子树 根 入栈
   8. 执行下一次循环
9. 插入排序
   1. 从I = 1位置开始 做外层循环
   2. 判断 如果 I -1 > I 将i指向的数据缓存
   3. j开始做内侧循环，找到要插入的位置 将缓存数据插入到 j +1位置上
   4. i++