EX4说明文档及学习心得

1. task1、3是一起写的，因为3可以用到1的中序遍历的功能打印树的所有结点。
2. task1不用递归的中序遍历就是用栈实现。由于父亲结点在左孩子之后访问，故要用栈将父亲结点暂时保存，于是在遍历左子树之前的步骤就是将当前结点入栈。注意这里有两个循环嵌套，外层循环是遍历整棵树，判断条件是p非空或者栈非空（p且栈空代表树遍历结束）；里层遍历是一直向下访问到最左的结点，开始往上走。访问当前结点右孩子“p=p->rightChild;”，如果右孩子为空再到 “p=stack.GetTop(); stack.Pop();”此时出栈的是当前结点的父结点。
3. task3由于是构造完全二叉树，就可以将结点存在数组中，再利用下标，i-th结点的左右孩子结点分别是2i+1-th和2i+2-th来实现即可。
4. task2和task4是一起写的，因为都用到层次遍历相关的内容。层次遍历用队列实现，主要思路是：当前结点出队列的同时把其左右孩子分别加入队列。
5. task2最重要的是检查输入，对字符串的操作要按照下标用switch分情况讨论，这里曾犯了一个错误，注意不能在switch的case里面定义某个变量，因为编译器会认为别的case访问不了而报错。因为要在switch之前声明，在不同case中再分别赋值。还有就是new完之后的delete，destroy函数也是纠结了很久，最后还是选择层次遍历的方式，不会让父亲结点被删除导致无法访问孩子结点。
6. 由于我让task4的用户输入也输入类似task2一样的序列，因此都和父亲结点有关系，所以都加了一个find函数用已知的父亲结点的data来返回对应的父亲结点，然后借用写过的中序遍历来遍历树并判断哪个是父亲结点并返回。
7. task4要实现遍历i-th层的结点，于是给结点类加一个level成员变量属性，作为后面层次遍历的开始和结束的判断条件。在increase函数调用前要用条件qu.GetFront()->GetLevel()来判断此时队列中最先要出队列的元素的level是不是指定的，必须在出队列前判断，否则就晚了，所以是用GetFront()。