在二叉查找树中增加一个成员函数，找出集合中第 i 大的元素。

【解】这个问题可以用递归的方法来解决。二叉查找树是一棵有序树，左子树上的所有结点值都小于根结点的值，右子树上的所有结点值都大于根结点的值。按照递归的观点，如果右子树的规模大于或等于 i 时，第 i 个大的元素存在于右子树，且为右子树上的第 i 个大的元素。如果右子树的规模等于 i-1，那么根结点就是整棵树上第 i 个大的元素。如果右子树的规模小于 i-1，第 i 个大的元素存在于左子树上，且为左子树上的第（i-1-右子树规模）个大的元素。如果有一个函数 size 可以计算以 t 为根的树的规模，那么找二叉查找树上第 i 个大的元素的函数 findKth 的函数的实现就可以用上述的递归过程完成，具体见代码清单 8-33。

代码清单 8-33 findmax 函数的实现

```
1.  template <class Type>                    // 公有的 findKth 函数
2.  bool BinarySearchTree<Type>::findKth (int i, Type &x) const
3.  {    return findKth (i, x, root); }
4.
5.  template <class Type>                    // 私有的 findKth 函数
6.  bool BinarySearchTree<Type>::findKth (int i, Type &x, BinaryNode *t) const
7.  {    if (t == NULL) return false;  // 空树，找不到
8.       int rs = size(t->right);
9.       if (rs == i - 1) {          // 根结点是第 i 大的元素
10.           x = t->data;
11.           return true;
12.      }
13.      if (rs >= i) return findKth (i, x, t->right);
14.      else returnfindKth (i-rs-1, x, t->left);
15. }
```

公有的 findKth 函数有两个参数：i 和 x，x 存放找到的第 i 个大的元素的值。函数返回一个 bool 值。当找到了第 i 个大的元素时返回 true，否则返回 false。公有的 findKth 函数调用私有的 findKth 函数完成自己的任务。私有的 findKth 函数比公有的 findKth 函数多一个参数：树根 t。它完全按照上述的递归过程实现。

还需要一个求树规模的函数 size。这个函数在二叉树类中介绍过。size 函数是二叉查找树类中私有成员函数，它的实现见代码清单 8-34。

代码清单 8-34 size 函数的实现

```
16. template <class Type>
17. int BinarySearchTree<Type>::size( BinaryNode *t) const
18. {
19.      if (t == NULL) return 0;
20.      return 1 + size(t->left) + size(t->right);
21. }
```