```
//折半查找

int Bsearch(int R[], int low, int high, int k){
    int mid;
    while(low <= high)  // 当子表长度大于等于1时进行循环
    {
        mid = (low + high) / 2;
        if(R[mid] == k)
            return mid;
        else if(R[mid] > k)
            high = mid - 1;
        else
            low = mid + 1;
    }
    return 0;
}

// 索引查找（分块查找）
// 第一步二分查找，第二步顺序查找
typedef struct{
    int key;         //假设表内元素为int型
    int low, high;  //记录某块中第一个和最后一个元素的位置
}indexElem;
indexElem index[maxSize]; // 定义索引表
```

```
//二叉排序树 查找
BTNode* BSTSearch(BTNode* bt, int key)
{
    if(bt == null)
        return null;
    else
    {
        if(bt->val == key)
            return bt;
        else if(key < bt->val)
            return BSTSearch(bt->left, key);
        else
            return BSTSearch(bt->right, key);
    }
}

//二叉排序树 插入
int BSTInsert(BTNode* bt, int key)
{
    if(bt == null)
    {
        BTNode* bt = new BTNode(key);
        return 1;
    }
    else
    {
        if(key == bt->val)
            return 0;
        else if(key < bt->val)
            BSTInsert(bt->left, key);
        else
            BStInsert(bt->right, key);
    }
}
```