```cpp
#include <iostream>

class TreeNode {
public:
    int key;
    TreeNode* left;
    TreeNode* right;

    TreeNode(int value) : key(value), left(nullptr), right(nullptr) {}
};

class BinarySearchTree {
private:
    TreeNode* root;

    TreeNode* insert(TreeNode* root, int key) {
        if (root == nullptr) {
            return new TreeNode(key);
        }

        if (key < root->key) {
            root->left = insert(root->left, key);
        } else if (key > root->key) {
            root->right = insert(root->right, key);
        }

        return root;
    }

    TreeNode* deleteNode(TreeNode* root, int key) {
        if (root == nullptr) {
            return root;
        }

        if (key < root->key) {
            root->left = deleteNode(root->left, key);
        } else if (key > root->key) {
            root->right = deleteNode(root->right, key);
        } else {
            if (root->left == nullptr) {
                TreeNode* temp = root->right;
                delete root;
                return temp;
            } else if (root->right == nullptr) {
                TreeNode* temp = root->left;
                delete root;
                return temp;
            }

            root->key = minValueNode(root->right);
            root->right = deleteNode(root->right, root->key);
        }

        return root;
    }

    int minValueNode(TreeNode* node) {
        TreeNode* current = node;
        while (current->left != nullptr) {
            current = current->left;
        }
        return current->key;
    }

    void inorderTraversal(TreeNode* root) {
        if (root != nullptr) {
            inorderTraversal(root->left);
            std::cout << root->key << " ";
            inorderTraversal(root->right);
        }
    }

public:
    BinarySearchTree() : root(nullptr) {}

    void insert(int key) {
        root = insert(root, key);
    }

    void deleteNode(int key) {
        root = deleteNode(root, key);
    }

    void inorderTraversal() {
        inorderTraversal(root);
        std::cout << std::endl;
    }
};

int main() {
    BinarySearchTree bst;

    int keys[] = {50, 30, 70, 20, 40, 60, 80};

    for (int key : keys) {
        bst.insert(key);
    }

```

```cpp
    std::cout << "In-order Traversal: ";
    bst.inorderTraversal();

    bst.deleteNode(50);
    std::cout << "In-order Traversal after deleting : ";
    bst.inorderTraversal();

    return 0;
}

```