화일처리 보고서
이진탐색트리

20181653 이강희

October 30, 2019

강의 슬라이드를 참조하였습니다.

1 출력 결과

Listing 1: output.txt

```
 File Processing
Binary Search Tree
==================
Insert 40       40
Insert 11       11 40
Insert 77       11 40 77
Insert 33       11 33 40 77
Insert 20       11 20 33 40 77
Insert 90       11 20 33 40 77 90
Insert 99       11 20 33 40 77 90 99
Insert 70       11 20 33 40 70 77 90 99
Insert 88       11 20 33 40 70 77 88 90 99
Insert 80       11 20 33 40 70 77 80 88 90 99
Insert 66       11 20 33 40 66 70 77 80 88 90 99
Insert 10       10 11 20 33 40 66 70 77 80 88 90 99
Insert 22       10 11 20 22 33 40 66 70 77 80 88 90 99
Insert 30       10 11 20 22 30 33 40 66 70 77 80 88 90 99
Insert 44       10 11 20 22 30 33 40 44 66 70 77 80 88 90 99
Insert 55       10 11 20 22 30 33 40 44 55 66 70 77 80 88 90 99
Insert 50       10 11 20 22 30 33 40 44 50 55 66 70 77 80 88 90 99
Insert 60       10 11 20 22 30 33 40 44 50 55 60 66 70 77 80 88 90 99
Insert 100      10 11 20 22 30 33 40 44 50 55 60 66 70 77 80 88 90 99 100
Delete 40       10 11 20 22 30 33 44 50 55 60 66 70 77 80 88 90 99 100
Delete 11       10 20 22 30 33 44 50 55 60 66 70 77 80 88 90 99 100
Delete 77       10 20 22 30 33 44 50 55 60 66 70 80 88 90 99 100
Delete 33       10 20 22 30 44 50 55 60 66 70 80 88 90 99 100
Delete 20       10 22 30 44 50 55 60 66 70 80 88 90 99 100
Delete 90       10 22 30 44 50 55 60 66 70 80 88 99 100
Delete 99       10 22 30 44 50 55 60 66 70 80 88 100
Delete 70       10 22 30 44 50 55 60 66 80 88 100
Delete 88       10 22 30 44 50 55 60 66 80 100
Delete 80       10 22 30 44 50 55 60 66 100
Delete 66       10 22 30 44 50 55 60 100
Delete 10       22 30 44 50 55 60 100
Delete 22       30 44 50 55 60 100
Delete 30       44 50 55 60 100
Delete 44       50 55 60 100
Delete 55       50 60 100
Delete 50       60 100
Delete 60       100
Delete 100
Insert 40       40
Insert 11       11 40
Insert 77       11 40 77
Insert 33       11 33 40 77
Insert 20       11 20 33 40 77
Insert 90       11 20 33 40 77 90
Insert 99       11 20 33 40 77 90 99
Insert 70       11 20 33 40 70 77 90 99
Insert 88       11 20 33 40 70 77 88 90 99
Insert 80       11 20 33 40 70 77 80 88 90 99
Insert 66       11 20 33 40 66 70 77 80 88 90 99
Insert 10       10 11 20 33 40 66 70 77 80 88 90 99
```

```
Insert 22       10 11 20 22 33 40 66 70 77 80 88 90 99
Insert 30       10 11 20 22 30 33 40 66 70 77 80 88 90 99
Insert 44       10 11 20 22 30 33 40 44 66 70 77 80 88 90 99
Insert 55       10 11 20 22 30 33 40 44 55 66 70 77 80 88 90 99
Insert 50       10 11 20 22 30 33 40 44 50 55 66 70 77 80 88 90 99
Insert 60       10 11 20 22 30 33 40 44 50 55 60 66 70 77 80 88 90 99
Insert 100      10 11 20 22 30 33 40 44 50 55 60 66 70 77 80 88 90 99 100
Delete 100      10 11 20 22 30 33 40 44 50 55 60 66 70 77 80 88 90 99
Delete 60       10 11 20 22 30 33 40 44 50 55 66 70 77 80 88 90 99
Delete 50       10 11 20 22 30 33 40 44 55 66 70 77 80 88 90 99
Delete 55       10 11 20 22 30 33 40 44 66 70 77 80 88 90 99
Delete 44       10 11 20 22 30 33 40 66 70 77 80 88 90 99
Delete 30       10 11 20 22 33 40 66 70 77 80 88 90 99
Delete 22       10 11 20 33 40 66 70 77 80 88 90 99
Delete 10       11 20 33 40 66 70 77 80 88 90 99
Delete 66       11 20 33 40 70 77 80 88 90 99
Delete 80       11 20 33 40 70 77 88 90 99
Delete 88       11 20 33 40 70 77 90 99
Delete 70       11 20 33 40 77 90 99
Delete 99       11 20 33 40 77 90
Delete 90       11 20 33 40 77
Delete 20       11 33 40 77
Delete 33       11 40 77
Delete 77       11 40
Delete 11       40
Delete 40
```

---

2 소스 코드

```cpp
#include <iostream>
using namespace std;

class TreeNode{
public:
    TreeNode(int key=0, TreeNode *left=NULL, TreeNode *right=NULL) : key(key), left(left), right(right) {}
    int key;
    TreeNode *left;
    TreeNode *right;
};

TreeNode *getNode(int key=0, TreeNode *left=NULL, TreeNode *right=NULL) {
    return new TreeNode(key, left, right);
}

bool insertBST(TreeNode *&T, int newKey) {
    if (T == NULL) {
        T = getNode(newKey);
        return true;
    }

    TreeNode *q = NULL;
    TreeNode *p = T;
    while (p != NULL) {
        if (newKey == p->key) return false;
        q = p;
```

```
        if (newKey < p->key) p = p->left;
        else p = p->right;
    }

    TreeNode *newNode = getNode(newKey);

    if (T == NULL) T = newNode;
    else if(newKey < q->key) q->left = newNode;
    else q->right = newNode;
    return true;
}

int height(TreeNode *T) {
    if (T == NULL) return 0;
    return max(height(T->left), height(T->right)) + 1;
}

int noNodes(TreeNode *T) {
    if (T == NULL) return 0;
    return noNodes(T->left) + noNodes(T->right) + 1;
}

TreeNode *maxNode(TreeNode *T) {
    if (T->right != NULL) return maxNode(T->right);
    return T;
}

TreeNode *minNode(TreeNode *T) {
    if (T->left != NULL) return minNode(T->left);
    return T;
}

bool deleteBST(TreeNode *&T, int deleteKey) {
    TreeNode *p = T;
    TreeNode *q = NULL;
    while (p->key != deleteKey) {
        if (p == NULL) return false;
        q = p;
        if (deleteKey < p->key) p = p->left;
        else p = p->right;
    }

    if (p->left == NULL && p->right == NULL) {
        if (q == NULL) T = NULL;
        else if (q->left == p) q->left = NULL;
        else q->right = NULL;
        delete p;
    }
    else if (p->left == NULL || p->right == NULL) {
        if (p->left != NULL) {
            if (q == NULL) T = p->left;
            else if (q->left == p) q->left = p->left;
            else q->right = p->left;
        }
        else {
            if (q == NULL) T = p->right;
```

```
            else if (q->left == p) q->left = p->right;
            else q->right = p->right;
        }
        delete p;
    }
    else {
        int FLAG = 0; // 0: LEFT, 1: RIGHT
        TreeNode *r;
        if (height(p->left) > height(p->right)) {
            r = maxNode(p->left);
            FLAG = 0;
        }
        else if (height(p->left) < height(p->right)) {
            r = minNode(p->right);
            FLAG = 1;
        }
        else {
            if (noNodes(p->left) >= noNodes(p->right)) {
                r = maxNode(p->left);
                FLAG = 0;
            }
            else {
                r = minNode(p->right);
                FLAG = 1;
            }
        }
        p->key = r->key;
        if (FLAG == 0) deleteBST(p->left, r->key);
        else deleteBST(p->right, r->key);
    }
    return true;
}

void inorder(TreeNode *T) {
    if (T == NULL) return;
    inorder(T->left);
    cout << T->key << ' ';
    inorder(T->right);
}

int main() {
    cout << " File Processing  " << endl;
    cout << "Binary Search Tree" << endl;
    cout << "==================" << endl;

    int keyArray[] = {40, 11, 77, 33, 20, 90, 99, 70, 88, 80, 66, 10, 22, 30, 44, 55, 50, 60, 100};

    TreeNode *root = NULL;

    for (int insertKey : keyArray) {
        cout << "Insert " << insertKey << "\t";
        insertBST(root, insertKey);
        inorder(root);
        cout << endl;
    }
```

```
    for (int deleteKey : keyArray) {
        cout << "Delete " << deleteKey << "\t";
        deleteBST(root, deleteKey);
        inorder(root);
        cout << endl;
    }

    root = NULL;

    for (int insertKey : keyArray) {
        cout << "Insert " << insertKey << "\t";
        insertBST(root, insertKey);
        inorder(root);
        cout << endl;
    }

    for (int i=18;i>=0;--i) {
        cout << "Delete " << keyArray[i] << "\t";
        deleteBST(root, keyArray[i]);
        inorder(root);
        cout << endl;
    }

    return 0;
}
```