資料結構第五次作業

班級:資工二乙

姓名:侯秉辰

學號:4B2G0116

一・說明

請寫一個 C++程式，完成以下要求：

1. 使用者輸入不等數量的整數值，分別產生 AVL tree 與 2-3 tree
2. 將所產生的結果以樹狀圖方式呈現出來
3. 針對所產生的樹，分別再提供插入與刪除某一資料的功能

二・程式

```
#include <iostream>
#include <vector>
#include <queue>
#include <iomanip>
#include <algorithm>

using namespace std;

class AVLNode {
public:
    int key;
    AVLNode* left;
    AVLNode* right;
    int height;

    AVLNode(int k) : key(k), left(nullptr), right(nullptr), height(1) {}
};

class AVLTree {
public:
    AVLNode* root;

    AVLTree() : root(nullptr) {}

    int height(AVLNode* node) {
        return node ? node->height : 0;
    }

    int balanceFactor(AVLNode* node) {
        return node ? height(node->left) - height(node->right) : 0;
    }
```

```
AVLNode* rotateRight(AVLNode* y) {
    AVLNode* x = y->left;
    AVLNode* T2 = x->right;

    x->right = y;
    y->left = T2;

    y->height = max(height(y->left), height(y->right)) + 1;
    x->height = max(height(x->left), height(x->right)) + 1;

    return x;
}

AVLNode* rotateLeft(AVLNode* x) {
    AVLNode* y = x->right;
    AVLNode* T2 = y->left;

    y->left = x;
    x->right = T2;

    x->height = max(height(x->left), height(x->right)) + 1;
    y->height = max(height(y->left), height(y->right)) + 1;

    return y;
}

AVLNode* insert(AVLNode* node, int key) {
    if (!node) return new AVLNode(key);

    if (key < node->key)
        node->left = insert(node->left, key);
    else if (key > node->key)
        node->right = insert(node->right, key);
    else
        return node;

    node->height = max(height(node->left), height(node->right)) + 1;
```

```
        int balance = balanceFactor(node);

        // Left Left
        if (balance > 1 && key < node->left->key)
            return rotateRight(node);

        // Right Right
        if (balance < -1 && key > node->right->key)
            return rotateLeft(node);

        // Left Right
        if (balance > 1 && key > node->left->key) {
            node->left = rotateLeft(node->left);
            return rotateRight(node);
        }

        // Right Left
        if (balance < -1 && key < node->right->key) {
            node->right = rotateRight(node->right);
            return rotateLeft(node);
        }

        return node;
    }

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

    void inOrder(AVLNode* node) {
        if (!node) return;
        inOrder(node->left);
        cout << node->key << " ";
        inOrder(node->right);
    }

    void inOrder() {
```

```
        inOrder(root);
        cout << endl;
    }

    void printTree(AVLNode* root, int space = 0, int height = 10) {
        if (!root) return;

        space += height;
        printTree(root->right, space);

        cout << endl;
        for (int i = height; i < space; i++)
            cout << " ";
        cout << root->key << "\n";

        printTree(root->left, space);
    }

    void printTree() {
        printTree(root);
    }
};

class TwoThreeTree {
public:
    struct Node {
        int keys[3];
        Node* children[4];
        int numKeys;
        bool isLeaf;

        Node() {
            fill(keys, keys + 3, 0);
            fill(children, children + 4, nullptr);
            numKeys = 0;
            isLeaf = true;
        }
    };
```

```
    Node* root;

    TwoThreeTree() : root(nullptr) {}

    void insert(int key) {
        // Placeholder for 2-3 tree insertion
        cout << "2-3 Tree insertion is not yet implemented.\n";
    }

    void deleteKey(int key) {
        // Placeholder for 2-3 tree deletion
        cout << "2-3 Tree deletion is not yet implemented.\n";
    }

    void printTree() {
        // Placeholder for 2-3 tree print
        cout << "2-3 Tree printing is not yet implemented.\n";
    }
};

int main() {
    AVLTree avl;
    TwoThreeTree twoThree;

    vector<int> numbers;
    int num;

    cout << "請輸入整數值（輸入 -1 結束）：\n";
    while (true) {
        cin >> num;
        if (num == -1) break;
        numbers.push_back(num);
    }

    // 建立 AVL 樹
    for (int n : numbers)
        avl.insert(n);
```

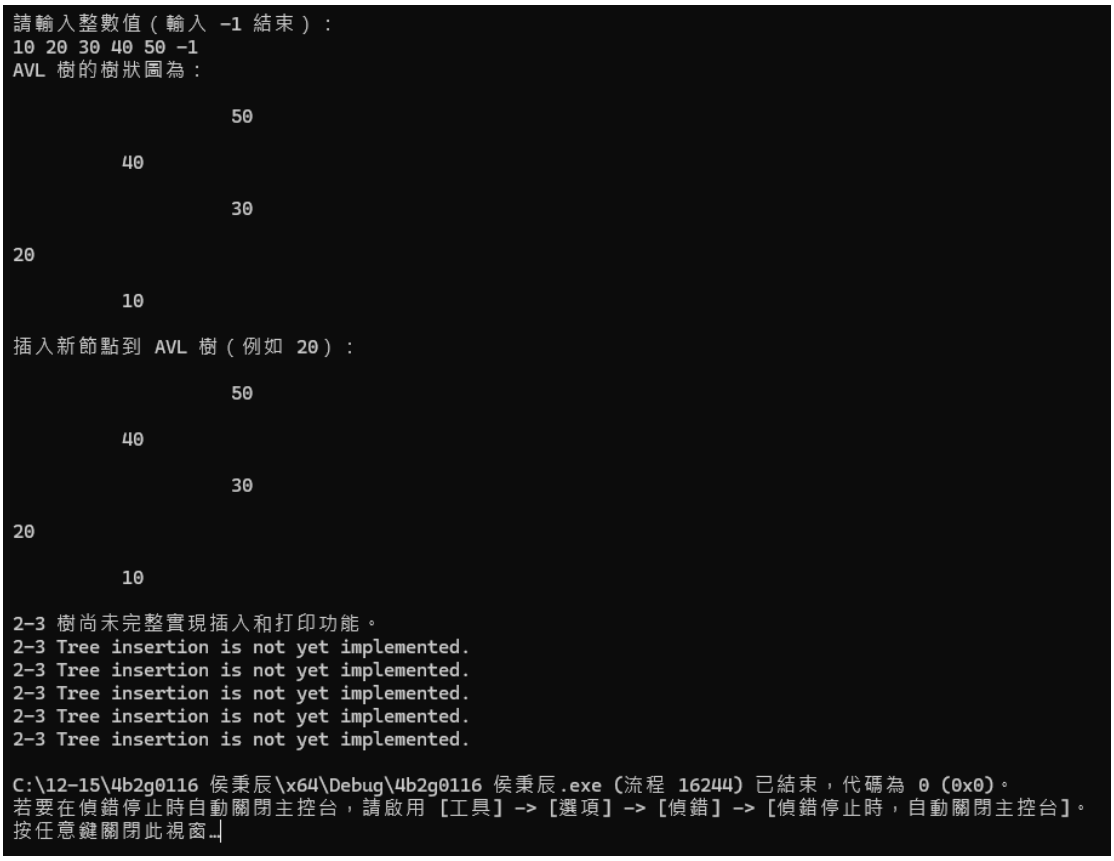
```
        // 顯示 AVL 樹
        cout << "AVL 樹的樹狀圖為：\n";
        avl.printTree();

        cout << "\n 插入新節點到 AVL 樹（例如 20）：\n";
        avl.insert(20);
        avl.printTree();

        // 建立 2-3 樹（尚未實現插入）
        cout << "\n2-3 樹尚未完整實現插入和打印功能。\n";
        for (int n : numbers)
            twoThree.insert(n);

        return 0;
    }
```

三・執行結果



https://github.com/4b2g0116/4B2G0116-.git