第六次作業

班級:資工二乙

姓名:侯秉辰

學號:4B2G0116

一・題目

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 m-way 搜尋樹和 B-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

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

二・程式與程式說明

```
#include <iostream>
#include <vector>
#include <queue>
#include <iomanip>
using namespace std;

// Node structure for M-Way Tree
struct MWayNode {
    vector<int> keys;
    vector<MWayNode*> children;
    bool isLeaf;

    MWayNode(int m) : isLeaf(true) {
        keys.reserve(m - 1);
        children.resize(m, nullptr);
    }
};

// Node structure for B-Tree
struct BTreeNode {
    vector<int> keys;
    vector<BTreeNode*> children;
    bool isLeaf;

    BTreeNode(bool leaf) : isLeaf(leaf) {}
};

// M-Way Tree Functions
MWayNode* insertMWay(MWayNode* root, int key, int m) {
    if (!root) {
        root = new MWayNode(m);
        root->keys.push_back(key);
        return root;
    }

    // Simple insertion logic for demonstration (no balancing)
    if (root->isLeaf) {
        root->keys.push_back(key);
        sort(root->keys.begin(), root->keys.end());
        return root;
    }

    // Find the child to recurse into
    int i = 0;
    while (i < root->keys.size() && key > root->keys[i]) i++;
    root->children[i] = insertMWay(root->children[i], key, m);
    return root;
}

MWayNode* deleteMWay(MWayNode* root, int key, int m) {
    if (!root) return nullptr;

    // Simple deletion logic for demonstration (no balancing)
    auto it = find(root->keys.begin(), root->keys.end(), key);
    if (it != root->keys.end()) {
        root->keys.erase(it);
        return root;
    }

    // Recurse into children
    int i = 0;
    while (i < root->keys.size() && key > root->keys[i]) i++;
    if (root->children[i])
        root->children[i] = deleteMWay(root->children[i], key, m);
    return root;
}

void visualizeMWayTree(MWayNode* root, int level = 0) {
    if (!root) return;
```

```
        cout << string(level * 4, ' ') << "|-- ";
        for (int k : root->keys) cout << k << " ";
        cout << endl;
        for (auto child : root->children) visualizeMWayTree(child, level + 1);
}

// B-Tree Functions
void insertBTree(BTreeNode*& root, int key, int t) {
        if (!root) {
                root = new BTreeNode(true);
                root->keys.push_back(key);
                return;
        }

        // Simple insertion logic for demonstration (no splitting)
        if (root->isLeaf) {
                root->keys.push_back(key);
                sort(root->keys.begin(), root->keys.end());
                return;
        }

        // Find the child to recurse into
        int i = 0;
        while (i < root->keys.size() && key > root->keys[i]) i++;
        insertBTree(root->children[i], key, t);
}

void deleteBTree(BTreeNode*& root, int key, int t) {
        if (!root) return;

        // Simple deletion logic for demonstration (no balancing)
        auto it = find(root->keys.begin(), root->keys.end(), key);
        if (it != root->keys.end()) {
                root->keys.erase(it);
                return;
        }

        // Recurse into children
        int i = 0;
        while (i < root->keys.size() && key > root->keys[i]) i++;
        if (root->children[i])
                deleteBTree(root->children[i], key, t);
}

void visualizeBTree(BTreeNode* root, int level = 0) {
        if (!root) return;
        cout << string(level * 4, ' ') << "|-- ";
        for (int k : root->keys) cout << k << " ";
        cout << endl;
        for (auto child : root->children) visualizeBTree(child, level + 1);
}

// Main Function
int main() {
        int m, t;
        cout << "Enter the value of m for the M-Way Tree: ";
        cin >> m;
        cout << "Enter the minimum degree t for the B-Tree: ";
        cin >> t;

        MWayNode* mWayRoot = nullptr;
        BTreeNode* bTreeRoot = nullptr;

        int choice;
        do {
                cout << "\nMenu:" << endl;
                cout << "1. Insert into M-Way Tree" << endl;
                cout << "2. Insert into B-Tree" << endl;
                cout << "3. Delete from M-Way Tree" << endl;
                cout << "4. Delete from B-Tree" << endl;
                cout << "5. Visualize M-Way Tree" << endl;
                cout << "6. Visualize B-Tree" << endl;
                cout << "7. Exit" << endl;
                cout << "Enter your choice: ";
                cin >> choice;
```

```
switch (choice) {
    case 1: {
        int key;
        cout << "Enter a key to insert into the M-Way Tree: ";
        cin >> key;
        mWayRoot = insertMWay(mWayRoot, key, m);
        break;
    }
    case 2: {
        int key;
        cout << "Enter a key to insert into the B-Tree: ";
        cin >> key;
        insertBTree(bTreeRoot, key, t);
        break;
    }
    case 3: {
        int key;
        cout << "Enter a key to delete from the M-Way Tree: ";
        cin >> key;
        mWayRoot = deleteMWay(mWayRoot, key, m);
        break;
    }
    case 4: {
        int key;
        cout << "Enter a key to delete from the B-Tree: ";
        cin >> key;
        deleteBTree(bTreeRoot, key, t);
        break;
    }
    case 5:
        cout << "\nVisualizing M-Way Tree:" << endl;
        visualizeMWayTree(mWayRoot);
        break;
    case 6:
        cout << "\nVisualizing B-Tree:" << endl;
        visualizeBTree(bTreeRoot);
        break;
    case 7:
```

```
                cout << "Exiting the program." << endl;
                break;
            default:
                cout << "Invalid choice! Try again." << endl;
        }
    } while (choice != 7);

    return 0;
}
```