**Lab Assignment No. 4(B)**

**Code:**

A book consists of chapters, chapters consist of sections and sections consist of subsections. Construct a tree and print the nodes. Find the time and space requirements of your method.

#include <iostream>

#include <string>

#include <vector>

#include <limits> // For numeric\_limits

using namespace std;

// Node structure for the book hierarchy

struct Node {

string label;

vector<Node\*> children; // Each node can have multiple children

};

// Class to manage the tree structure

class BookTree {

private:

Node\* root; // Root node representing the book

public:

BookTree() { root = nullptr; }

void createTree();

void display(Node\* node, int depth = 0);

Node\* getRoot() { return root; }

~BookTree() { deleteTree(root); } // Destructor to free memory

void deleteTree(Node\* node);

};

// Function to safely get a positive integer input

int getValidInteger(string prompt) {

int num;

while (true) {

cout << prompt;

cin >> num;

if (cin.fail() || num < 0) {

cin.clear(); // Clear error flag

cin.ignore(numeric\_limits<streamsize>::max(), '\n'); // Ignore invalid input

cout << "Invalid input! Please enter a valid positive integer.\n";

} else {

cin.ignore(); // Clear newline character from input buffer

return num;

}

}

}

// Function to create the book hierarchy

void BookTree::createTree() {

root = new Node;

cout << "Enter the name of the book: ";

getline(cin, root->label);

int numChapters = getValidInteger("Enter the number of chapters in the book: ");

root->children.resize(numChapters);

for (int i = 0; i < numChapters; i++) {

root->children[i] = new Node;

cout << "Enter the name of Chapter " << i + 1 << ": ";

getline(cin, root->children[i]->label);

int numSections = getValidInteger("Enter the number of sections in Chapter " + root->children[i]->label + ": ");

root->children[i]->children.resize(numSections);

for (int j = 0; j < numSections; j++) {

root->children[i]->children[j] = new Node;

cout << "Enter the name of Section " << j + 1 << ": ";

getline(cin, root->children[i]->children[j]->label);

int numSubsections = getValidInteger("Enter the number of subsections in Section " + root->children[i]->children[j]->label + ": ");

root->children[i]->children[j]->children.resize(numSubsections);

for (int k = 0; k < numSubsections; k++) {

root->children[i]->children[j]->children[k] = new Node;

cout << "Enter the name of Subsection " << k + 1 << ": ";

getline(cin, root->children[i]->children[j]->children[k]->label);

}

}

}

}

// Recursive function to display the book hierarchy

void BookTree::display(Node\* node, int depth) {

if (node == nullptr) return;

// Print indentation based on depth

for (int i = 0; i < depth; i++)

cout << " ";

cout << "- " << node->label << endl;

// Recursively display children

for (Node\* child : node->children) {

display(child, depth + 1);

}

}

// Function to delete the tree and free memory

void BookTree::deleteTree(Node\* node) {

if (node == nullptr) return;

for (Node\* child : node->children) {

deleteTree(child);

}

delete node; // Free memory

}

// Main function to interact with the user

int main() {

BookTree bookTree;

int choice;

while (true) {

cout << "\n-----------------\n";

cout << "Book Tree Creation\n";

cout << "-----------------\n";

cout << "1. Create\n";

cout << "2. Display\n";

cout << "3. Quit\n";

cout << "Enter your choice: ";

cin >> choice;

cin.ignore(); // Clear newline from input buffer

switch (choice) {

case 1:

bookTree.createTree();

break;

case 2:

cout << "\n----- Book Hierarchy -----\n";

bookTree.display(bookTree.getRoot());

break;

case 3:

cout << "Thanks for using this program!\n";

return 0;

default:

cout << "Invalid choice! Please enter a valid option.\n";

}

}

}