Практична робота №5
Завірюха Еліна, МП-21

```
#include <iostream>
#include <queue>

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

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

class BinaryTree {
private:
    TreeNode* root;

    // Створення нового вузла з заданим значенням
    TreeNode* createNode(int value) {
        return new TreeNode(value);
    }

    // Запит на введення значення для лівого або правого нащадка
    std::string promptForValue(int nodeValue, const std::string& childType) {
        std::cout << "Введіть значення для " << childType << " нащадка " << nodeValue << " (або '0', якщо немає): ";
        std::string input;
        std::cin >> input;
        return input;
    }

    // Прямий обхід дерева
    void preorderTraversal(TreeNode* node) {
        if (node) {
            std::cout << node->value << " ";
            preorderTraversal(node->left);
            preorderTraversal(node->right);
        }
    }

    // Симетричний обхід дерева
    void inorderTraversal(TreeNode* node) {
        if (node) {
            inorderTraversal(node->left);
            std::cout << node->value << " ";
            inorderTraversal(node->right);
        }
    }

    // Зворотній обхід дерева
    void postorderTraversal(TreeNode* node) {
        if (node) {
            postorderTraversal(node->left);
            postorderTraversal(node->right);
            std::cout << node->value << " ";
        }
    }

    // Рекурсивна функція для підрахунку вузлів із заданим значенням
    int countNodesWithValue(TreeNode* node, int targetValue) {
        if (node == nullptr) {
            return 0;
        }

        int count = (node->value == targetValue) ? 1 : 0;
        count += countNodesWithValue(node->left, targetValue);
        count += countNodesWithValue(node->right, targetValue);

        return count;
    }

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

    // Створення двохпроходового бінарного дерева
    void createBinaryTree() {
        int rootValue;
        std::cout << "Введіть значення кореневого вузла: ";
        std::cin >> rootValue;
        root = createNode(rootValue);
```

```
        std::queue<TreeNode*> nodeQueue;
        nodeQueue.push(root);

        while (!nodeQueue.empty()) {
            TreeNode* currentNode = nodeQueue.front();
            nodeQueue.pop();

            std::string leftValue = promptForValue(currentNode->value, "лівого");
            currentNode->left = (leftValue != "0") ? createNode(std::stoi(leftValue)) : nullptr;
            if (currentNode->left) {
                nodeQueue.push(currentNode->left);
            }

            std::string rightValue = promptForValue(currentNode->value, "правого");
            currentNode->right = (rightValue != "0") ? createNode(std::stoi(rightValue)) : nullptr;
            if (currentNode->right) {
                nodeQueue.push(currentNode->right);
            }
        }
    }

    // Обхід дерева та підрахунок вузлів зі значенням
    void traverseAndCount() {
        std::cout << "Прямий обхід: ";
        preorderTraversal(root);
        std::cout << "\nСиметричний обхід: ";
        inorderTraversal(root);
        std::cout << "\nЗворотній обхід: ";
        postorderTraversal(root);

        int targetValue;
        std::cout << "\nВведіть значення, для якого потрібно знайти кількість вузлів: ";
        std::cin >> targetValue;

        int count = countNodesWithValue(root, targetValue);
        std::cout << "Кількість вузлів зі значенням " << targetValue << ": " << count << std::endl;
    }
};

int main() {
    BinaryTree binaryTree;
    binaryTree.createBinaryTree();
    binaryTree.traverseAndCount();

    return 0;
}
```