Министерство науки и высшего образования Российской Федерации
Калужский филиал
федерального государственного бюджетного
образовательного учреждения высшего образования
«Московский государственный технический университет имени Н.Э. Баумана (национальный исследовательский университет)»
(КФ МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ *ИУК "Информатика и управление"*

КАФЕДРА *ИУК5 "Системы обработки информации"*

ЛАБОРАТОРНАЯ РАБОТА №7

«Бинарные деревья»

ДИСЦИПЛИНА: «Вычислительные алгоритмы»

Выполнил: студент гр.ИУК5-41Б _____ (____Шиндин А.О.___)
(Подпись) (Ф.И.О.)

Проверил: _____ (____ Вершинин В.Е__)
(Подпись) (Ф.И.О.)

Дата сдачи (защиты):

Результаты сдачи (защиты):

- Балльная оценка:
- Оценка:

Калуга, 2023

Цель: получение практических навыков реализации бинарных деревьев.

Задачи:

1) Опишите класс — дерево, необходимое для решения задачи, указанной в вашем варианте задания, и реализуйте его методы.
2) Продемонстрируйте работу основных методов работы с деревом: построение, вывод, обход.
3) Составьте программу решения задачи, указанной в вашем варианте задания.

Задание для варианта 13:

На основе процедуры обхода дерева снизу вверх реализовать операцию поиска узла с заданным значением в дереве, не являющемся деревом поиска. Из двух последовательностей символов построить два бинарных дерева минимальной высоты. В первом дереве найти элемент с заданным значением и подключить второе дерево в качестве его левого поддерева, если оно пусто, или левого поддерева первого из его крайних левых потомков, имеющих пустое левое поддерево.

Ход работы:

Программа написанная на языке c++:

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

using namespace std;

// Структура для узла бинарного дерева
struct Node {
int data;
Node* left;
Node* right;

// Конструктор для узла
Node(int value) : data(value), left(nullptr), right(nullptr) {}

// Метод для вывода дерева с отступами
void printIndented(int indent = 0) {
if (right != nullptr) {
right->printIndented(indent + 4);
```

```
}
cout << string(indent, ' ') << data << endl;
if (left != nullptr) {
left->printIndented(indent + 4);
}
}
};

// Функция для построения сбалансированного дерева из вектора,
// принимая первое число в качестве корня
Node* buildBalancedTree(const vector<int>& nums, int start, int end) {
if (start > end) {
return nullptr;
}

// Используем первый элемент как корень
int rootIndex = start + (end - start) / 2;
Node* root = new Node(nums[rootIndex]);

// Рекурсивно строим левое и правое поддеревья
root->left = buildBalancedTree(nums, start, rootIndex - 1);
root->right = buildBalancedTree(nums, rootIndex + 1, end);

return root;
}

// Функция для присоединения второго дерева к первому
void attachSecondTree(Node* root, int value, Node* tree) {
// Ищем узел с заданным значением
queue<Node*> q;
q.push(root);
Node* targetNode = nullptr;

while (!q.empty()) {
Node* current = q.front();
q.pop();
```

```
if (current->data == value) {
targetNode = current;
break;
}

if (current->left) q.push(current->left);
if (current->right) q.push(current->right);
}

if (!targetNode) {
cout << "Node with the given value is not found in the tree." << endl;
return;
}

if (!targetNode->left) {
targetNode->left = tree;
} else {
Node* current = targetNode->left;
while (current->left) {
current = current->left;
}
current->left = tree;
}
}

int main() {
// Ввод чисел для первого дерева
cout << "Enter sequence of numbers for the first tree (separated by spaces): ";
vector<int> nums1;
int num;
while (cin >> num) {
nums1.push_back(num);
if (cin.peek() == '\n') break;
}

// Построение сбалансированного дерева из введенной последовательности для первого дерева
```

```
Node* firstTree = buildBalancedTree(nums1, 0, nums1.size() - 1);
firstTree->printIndented();

// Ввод чисел для второго дерева
cout << "Enter sequence of numbers for the second tree (separated by spaces): ";
vector<int> nums2;
while (cin >> num) {
nums2.push_back(num);
if (cin.peek() == '\n') break;
}

// Построение сбалансированного дерева из введенной последовательности для второго дерева
Node* secondTree = buildBalancedTree(nums2, 0, nums2.size() - 1);
secondTree->printIndented();

// Присоединяем второе дерево к первому
cout << "Enter numbers for attach Second Tree: ";
int numbers ={};
cin >> numbers;
attachSecondTree(firstTree, numbers, secondTree);

// Выводим результат
cout << "The first tree after attaching the second tree:" << endl;
firstTree->printIndented();

return 0;
}
```

Результат:

Вывод: в результате выполнения лабораторной работы были приобретены практичекие навыки реализации бинарных деревьев.