**МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**

# Лабораторная работа №6

по дисциплине «Программирование»

Деревья

Группа: **АВТ-008**

Студент: **Казанцев К.О. , Павлюк А.С.**

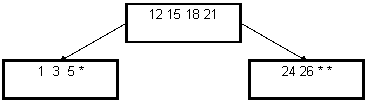
Преподаватель: **Гриньков Д.К.**

НОВОСИБИРСК 2021

## Задание

Программа должна содержать функцию  обхода дерева с выводом  его  содержимого,  функцию  добавления  вершины дерева  (ввод),  а  также  указанную  в  варианте  функцию.

1.      Вершина двоичного дерева содержит  массив  целых  и  два указателя на правое  и  левое  поддерево.  Массив  целых  в каждом элементе упорядочен, дерево  в  целом  также  упорядочено.  Функция  включает  в  дерево  целую  переменную  с сохранением упорядоченности.



## Проектирование программы

Обсуждение основных идей алгоритма

Идея: создать структуру бинарного дерева и очереди, функцию добавления элементов, функцию вывода с помощью очереди.

## Составные части программы

## 1. Структура бинарного дерева

struct btree {

int n;

int A[4];

struct btree\* left;

struct btree\* right;

## };

## 2. Структура очереди

struct queue {

btree\* tree;

int lvl;

queue\* next;

## };

## 3. Функция вывода с использованием очереди

|  |
| --- |
| void tree\_print(queue\* pb, queue\* pe) { |
| if (pb != NULL) { | if (pb != NULL) { |
| queue\* q = pb; | queue\* q = pb; |
| cout << q->lvl << ") "; | cout << q->lvl << ") "; |
| for (int i = 0; i < q->tree->n; i++) | for (int i = 0; i < q->tree->n; i++) |
| cout << q->tree->A[i] << " "; | cout << q->tree->A[i] << " "; |
| cout << endl; | cout << endl; |
| if (q->tree->left != NULL) { | if (q->tree->left != NULL) { |

## 3. Функция добавления элемента в дерево

void add(int b, btree\*& tree) {

if (tree == NULL) { //если нет узла

tree = new btree();

tree->A[0] = b;

tree->n++;

}

else if (tree->n < 4) { //если узел не заполнен

tree->A[tree->n] = b;

tree->n++;

for (int i = 0; i < tree->n - 1; i++) //сортировка

for (int j = 0; j < tree->n - i - 1; j++)

if (tree->A[j] > tree->A[j + 1]) {

int c = tree->A[j]; tree->A[j] = tree->A[j + 1]; tree->A[j + 1] = c;

}

}

else if (tree->n == 4) { //если узел заполнен

if (b >= tree->A[3]) add(b, tree->right); //если больше правого элемента узла

else if (b <= tree->A[0]) add(b, tree->left); //если меньше левого элемента узла

else if (b > tree->A[1]) { //если правее центра узла

int c = tree->A[3]; tree->A[3] = b; b = c; //замена

for (int i = 0; i < tree->n - 1; i++) //сортировка

for (int j = 0; j < tree->n - i - 1; j++)

if (tree->A[j] > tree->A[j + 1]) {

int c = tree->A[j]; tree->A[j] = tree->A[j + 1]; tree->A[j + 1] = c;

}

add(b, tree->right);

}

else if (b <= tree->A[1]) { //если левее центра узла

int c = tree->A[0]; tree->A[0] = b; b = c; //замена

for (int i = 0; i < tree->n - 1; i++) //сортировка

for (int j = 0; j < tree->n - i - 1; j++)

if (tree->A[j] > tree->A[j + 1]) {

int c = tree->A[j]; tree->A[j] = tree->A[j + 1]; tree->A[j + 1] = c;

}

add(b, tree->left);

}

}

## }

## Вывод

Программу можно улучшить, сбалансировав дерево, чтобы избежать большой разницы между высотами левого и правого поддеревьев.