МИНОБРНАУКИ РОССИИ

ФЕДЕРАЛЬАНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ

ВЫСШЕГО ОБРАЗОВАНИЯ

«ЮЖНЫЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

ИНЖЕНЕРНО-ТЕХНОЛОГИЧЕСКАЯ АКАДЕМИЯ

Институт компьютерных технологий и информационной безопасности

Кафедра высшей математики

**Отчёт по лабораторной работе №10**

по курсу «Алгоритмизация и программирование»

**на тему: «Бинарные деревья»**

Выполнил:

студент гр. КТбо1-1

Бураков А. В.

Проверил: ст.преподаватель каф. ИАСБ

Барковский С. А.

Таганрог 2024

**Цель:** Цель данной лабораторной работы привить необходимые навыки работы с такими структурами данных как деревья. Для успешного ее выполнения необходимо иметь представление о бинарных деревьях и их атрибутах: корнях, листьях, вершинах, высоте. Кроме того, необходимо уметь выполнять обход дерева и иметь представление о рекурсии и рекурсивном обходе дерева.

**Задачи**: В данной лабораторной работе необходимо написать программу, которая создает и заполняет двоичное дерево произвольными данными.

**Описание варианта:** Обойти двоичное дерево циклически и последовательно вывести значения каждой вершины на экран.

**Реализация**

#include <iostream>

#include <vector>

struct Node {

char data;

Node\* left;

Node\* right;

Node(char value) : data(value), left(nullptr), right(nullptr) {}

};

class Tree {

Node\* root;

Node\* insertRec(Node\* root, int value) {

if (root == nullptr) {

root = new Node(value);

return root;

}

if (rand() > RAND\_MAX / 2) {

if (root->left == nullptr) {

root->left = new Node(value);

}

else {

root->left = insertRec(root->left, value);

}

}

else {

if (root->right == nullptr) {

root->right = new Node(value);

}

else {

root->right = insertRec(root->right, value);

}

}

return root;

}

std::vector<std::vector<char>> printRec(Node\* p, int level, std::vector<std::vector<char>>\* arr) {

if (p) {

(\*arr)[level].push\_back(p->data);

printRec(p->left, level + 1, arr);

printRec(p->right, level + 1, arr);

}

else {

(\*arr)[level].push\_back(' ');

}

return \*arr;

}

void inorderTraversal(Node\* root) {

if (root == nullptr) {

return;

}

inorderTraversal(root->left);

std::cout << root->data << " ";

inorderTraversal(root->right);

}

void preorderTraversal(Node\* root) {

if (root == nullptr) {

return;

}

std::cout << root->data << " ";

preorderTraversal(root->left);

preorderTraversal(root->right);

}

void postorderTraversal(Node\* root) {

if (root == nullptr) {

return;

}

postorderTraversal(root->left);

postorderTraversal(root->right);

std::cout << root->data << " ";

}

public:

Tree() : root(nullptr) {}

void insert(int value) {

root = insertRec(root, value);

}

void print() {

std::vector<std::vector<char>> array(10);

array = printRec(root, 0, &array);

int prob = 8; //New abzac

int inter = 2; //Interval mesdu char

for (int i = 0; i < 10; ++i) {

for (int h = 0; h < prob; ++h)

std::cout << ' ';

for (int j = 0; j < size(array[i]); ++j) {

std::cout << array[i][j];

if (j % 2 == 0) {

for (int h = 0; h < inter + 2 - i; ++h)

std::cout << ' ';

}

else {

std::cout << ' ';

}

}

std::cout << "\n";

prob = prob - 2;

}

}

void inorder() {

inorderTraversal(root);

}

void preorder() {

preorderTraversal(root);

}

void postorder() {

postorderTraversal(root);

}

};

int main() {

Tree bst;

bst.insert('h');

bst.insert('j');

bst.insert('k');

bst.insert('l');

bst.insert('m');

bst.insert('c');

bst.insert('g');

bst.insert('a');

bst.insert('d');

std::cout << "Inorder traversal: ";

bst.inorder();

std::cout << "\n";

std::cout << "Preorder traversal: ";

bst.preorder();

std::cout << "\n";

std::cout << "Postorder traversal: ";

bst.postorder();

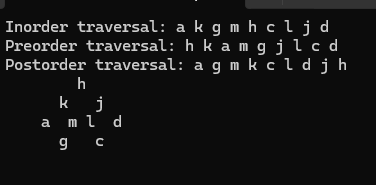
std::cout << "\n";

bst.print();

return 0;

}

**Результат работы**



**Вывод:** В данной лабораторной работе я выработал практические навыки по работе с такими структурами данных как деревья. Узнал информацию о бинарных деревьях и их атрибутах: корнях, листьях, вершинах, высоте. Кроме того, узнал как выполнять обход дерева и рекурсивный обход дерева. Написал программу, которая создает и заполняет двоичное дерево произвольными данными.