МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное

учреждение высшего образования

«ЧЕРЕПОВЕЦКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

|  |
| --- |
| Институт информационных технологий |
| Кафедра математического и программного обеспечения |
| Структуры и алгоритмы обработки данных |
|  |

ЛАБОРАТОРНАЯ РАБОТА №2

|  |  |
| --- | --- |
| Исполнитель: | студент  группы 1ПИб-02-3оп-23  Богданов  Ренат Алексеевич |
| Руководитель: | Журавлёва Юлия Михайловна |
| Оценка: |  |
| Подпись: |  |

2024 год

Задача

Написать функцию формирования бинарного дерева, состоящего из целых чисел. Для представления дерева использовать динамические структуры данных. Количество элементов дерева, а также его вид задаются случайным образом. Произвести вывод элементов дерева тремя видами обхода. Используя информацию о выведенном дереве изобразить структуру одного из построенных деревьев в отчете. Выполнить по вариантам следующие задания:

19. Написать рекурсивную функцию, которая находит минимальное и максимальное значения среди внутренних вершин дерева (см. рис 1).

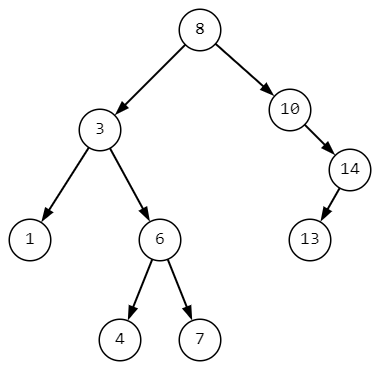


Рис. 1. Бинарное дерево

#include <iostream>

#include <limits> // Для INT\_MAX и INT\_MIN

struct Node

{

int data;

Node\* left, \* right;

};

// Функция для добавления узла в дерево

void add(Node\*& node, int value)

{

// Если узел пуст, создаем новый узел и инициализируем его

if (node == NULL)

{

node = new Node;

node->data = value;

node->left = NULL;

node->right = NULL;

}

// В противном случае рекурсивно добавляем узел в левое или правое поддерево

else if (value < node->data)

{

add(node->left, value);

}

else

{

add(node->right, value);

}

}

// Функция для печати дерева в виде списка узлов (префиксный обход)

void print(Node\* node)

{

if (node != NULL)

{

std::cout << node->data << " "; // Печатаем значение узла

print(node->left); // Рекурсивно печатаем левое поддерево

print(node->right); // Рекурсивно печатаем правое поддерево

}

}

// Функция для поиска минимального и максимального значений среди внутренних узлов дерева

void findMinMaxInternal(Node\* node, int& min, int& max) {

// Базовый случай: если узел пуст, возвращаемся

if (node == NULL) {

return;

}

// Если узел имеет хотя бы одного потомка, то это внутренний узел

if (node->left != NULL || node->right != NULL) {

// Обновляем минимальное значение, если текущий узел меньше текущего минимума

if (node->data < min) {

min = node->data;

}

// Обновляем максимальное значение, если текущий узел больше текущего максимума

if (node->data > max) {

max = node->data;

}

}

// Рекурсивно вызываем функцию для левого и правого поддеревьев

findMinMaxInternal(node->left, min, max);

findMinMaxInternal(node->right, min, max);

}

// 1. Предварительный обход (Pre-order)

void preOrder(Node\* node) {

if (node != NULL) {

std::cout << node->data << " "; // Печатаем значение узла

preOrder(node->left); // Рекурсивно вызываем функцию для левого поддерева

preOrder(node->right); // Рекурсивно вызываем функцию для правого поддерева

}

}

// 2. Симметричный обход (In-order)

void inOrder(Node\* node) {

if (node != NULL) {

inOrder(node->left); // Рекурсивно вызываем функцию для левого поддерева

std::cout << node->data << " "; // Печатаем значение узла

inOrder(node->right); // Рекурсивно вызываем функцию для правого поддерева

}

}

// 3. Последовательный обход (Post-order)

void postOrder(Node\* node) {

if (node != NULL) {

postOrder(node->left); // Рекурсивно вызываем функцию для левого поддерева

postOrder(node->right); // Рекурсивно вызываем функцию для правого поддерева

std::cout << node->data << " "; // Печатаем значение узла

}

}

// Функция для печати дерева в виде дерева (с отступами)

void printTree(Node\* node, int level = 0) {

if (node != NULL) {

printTree(node->right, level + 1); // Рекурсивно печатаем правое поддерево

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

std::cout << " "; // Печатаем отступы для визуализации уровня

}

std::cout << node->data << std::endl; // Печатаем значение узла

printTree(node->left, level + 1); // Рекурсивно печатаем левое поддерево

}

}

int main()

{

setlocale(LC\_ALL, "RU"); // Установка локали для корректного отображения кириллицы

Node\* root = NULL; // Инициализация корня дерева

// Добавление узлов в дерево

add(root, 8);

add(root->left, 3);

add(root->right, 10);

add(root->left->left, 1);

add(root->left->right, 6);

add(root->left->right->left, 4);

add(root->left->right->right, 7);

add(root->right->right, 14);

add(root->right->right->right, 13);

// Инициализация минимального и максимального значений

int min = std::numeric\_limits<int>::max();

int max = std::numeric\_limits<int>::min();

// Поиск минимального и максимального значений среди внутренних узлов

findMinMaxInternal(root, min, max);

// Вывод результатов

std::cout << "Minimal: " << min << std::endl;

std::cout << "Maximum: " << max << std::endl << std::endl;

// Вывод обходов дерева

std::cout << "Предварительный обход: ";

preOrder(root);

std::cout << std::endl;

std::cout << "Симметричный обход: ";

inOrder(root);

std::cout << std::endl;

std::cout << "Последовательный обход: ";

postOrder(root);

std::cout << std::endl;

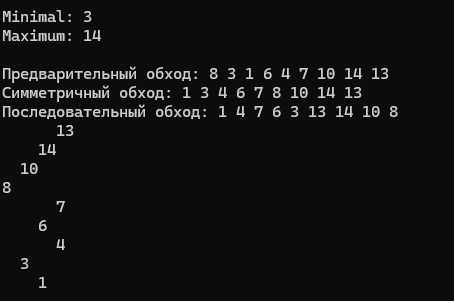
// Вывод дерева в виде дерева

printTree(root);

return 0;

}

Примеры работы программы



Вывод

В ходе выполнения лабораторной работы были освоены работа со структурой данных типа бинарное дерево и работа с рекуррентным алгоритмом прохода по дереву.