**САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ «ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)**

**Кафедра АПУ**

**ОТЧЕТ**

**по лабораторной работе №8**

**по Программированию**

Выполнил: Тихонов А.А

Группа №3391

Преподаватель: Писарев А.С

Санкт-Петербург

2024

**Описание задания:**

Расширить функциональность программы из лабораторной работы №6 путем ее

дополнения функцией вывода элементов построенного дерева на экран по уровням

**Реализация:**

Программа полностью соответствует программе лабораторной работы №7, за исключением добавления и последующего вызова следующих функций:

1. Рекурсивная функция для вычисления высоты дерева:

int tree\_height(Node \*ptr)

{

if (ptr == nullptr) {return 0;}

int left\_height = 0;

int right\_height = 0;

if (ptr -> left != nullptr) {left\_height = tree\_height(ptr -> left);}

if (ptr -> right != nullptr) {right\_height = tree\_height(ptr -> right);}

return max(left\_height, right\_height) + 1;

}

1. Функция для вывода конкретного уровня дерева:

void vyvod(Node \*ptr, const int level)

{

if (ptr == nullptr) {return;}

if (level == 1)

{

cout<<ptr -> n << " " <<endl;

}

else if (level > 1)

{

vyvod(ptr -> left, level -1);

vyvod(ptr -> right, level -1);

}

}

Функция вызывается внутри цикла for(), который выполняется nраз, где n– количество уровней дерева:

int height = tree\_height(ptr);

wcout<<L"Вывод дерева:" <<endl;

ptr = first;

for (int i = 1; i<= height; i++)

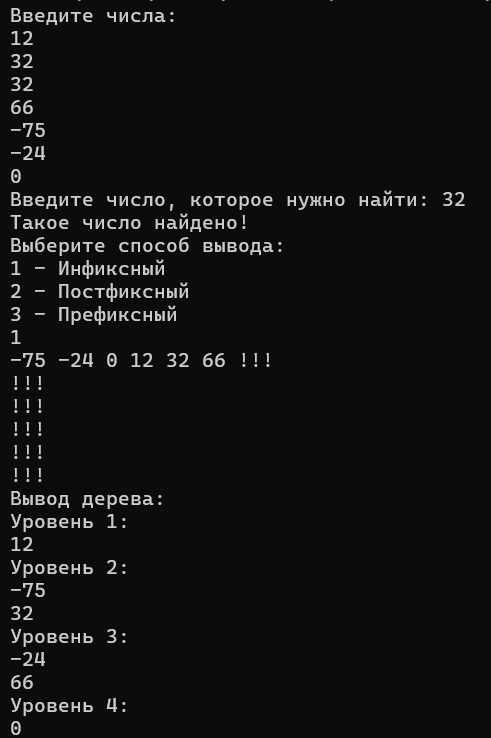
{

wcout<< L"Уровень " <<i<< ":" <<endl;

vyvod(ptr, i);

}

**Пример работы программы:**

****

**Вывод:**

Выполнив данную работу, я ознакомился со способами вывода бинарного дерева по уровням в языке С++

**Код программы:**

**#include <iostream>**

**#include <cstdlib>**

**#include <string>**

**#include <windows.h>**

**using namespace std;**

**struct Node**

**{**

**int n;**

**struct Node \*left;**

**struct Node \*right;**

**};**

**void add\_Node(Node \*ptr, int x, bool flag)**

**{**

**Node \*newNode = new Node;**

**newNode -> n = x;**

**newNode -> left = nullptr;**

**newNode -> right = nullptr;**

**if (flag)**

**{**

**ptr -> right = newNode;**

**}**

**if (!flag)**

**{**

**ptr -> left = newNode;**

**}**

**}**

**void delete\_tree(Node \*ptr)**

**{**

**if (ptr -> left != nullptr) {delete\_tree(ptr -> left);}**

**if (ptr -> right != nullptr) {delete\_tree(ptr -> right);}**

**delete ptr;**

**}**

**void find(Node \*ptr, int x)**

**{**

**bool flag = false;**

**while (true)**

**{**

**if (x == ptr -> n)**

**{**

**flag = true;**

**break;**

**}**

**if (x > ptr -> n)**

**{**

**if (ptr -> right != nullptr) {ptr = ptr -> right;}**

**else {break;}**

**}**

**if (x < ptr -> n)**

**{**

**if (ptr -> left != nullptr) {ptr = ptr -> left;}**

**else {break;}**

**}**

**}**

**if (flag) {wcout << L"Такоечислонайдено!" <<endl;}**

**else {wcout<<L"Такое число не найдено!" << endl;}**

**}**

**void prefix(Node\* ptr)**

**{**

**cout << ptr->n << " ";**

**if (ptr->left != nullptr)**

**{**

**prefix(ptr->left);**

**}**

**if (ptr->right != nullptr)**

**{**

**prefix(ptr->right);**

**}**

**}**

**void infix(Node\* ptr)**

**{**

**if (ptr->left != nullptr)**

**{**

**infix(ptr->left);**

**}**

**cout << ptr->n << " ";**

**if (ptr->right != nullptr)**

**{**

**infix(ptr->right);**

**}**

**}**

**void postfix(Node\* ptr)**

**{**

**if (ptr->left != nullptr)**

**{**

**postfix(ptr->left);**

**}**

**if (ptr->right != nullptr)**

**{**

**postfix(ptr->right);**

**}**

**cout << ptr->n << " ";**

**}**

**void vyvod(Node \*ptr, const int level)**

**{**

**if (ptr == nullptr) {return;}**

**if (level == 1)**

**{**

**cout << ptr -> n << " " << endl;**

**}**

**else if (level > 1)**

**{**

**vyvod(ptr -> left, level -1);**

**vyvod(ptr -> right, level -1);**

**}**

**}**

**int tree\_height(Node \*ptr)**

**{**

**if (ptr == nullptr) {return 0;}**

**int left\_height = 0;**

**int right\_height = 0;**

**if (ptr -> left != nullptr) {left\_height = tree\_height(ptr -> left);}**

**if (ptr -> right != nullptr) {right\_height = tree\_height(ptr -> right);}**

**return max(left\_height, right\_height) + 1;**

**}**

**int main()**

**{**

**setlocale(LC\_ALL, "RUSSIAN");**

**int a;**

**Node \*first = nullptr;**

**wcout<<L"Ввeдите числа:" <<endl;**

**cin >> a;**

**first = new Node;**

**first -> left = nullptr;**

**first -> right = nullptr;**

**first -> n = a;**

**Node \*ptr = first;**

**while (a != 0)**

**{**

**cin >> a;**

**while (true)**

**{**

**if (a > ptr -> n)**

**{**

**if (ptr -> right != nullptr)**

**{**

**ptr = ptr -> right;**

**} else**

**{**

**add\_Node(ptr, a, true);**

**break;**

**}**

**}**

**if (a < ptr -> n)**

**{**

**if (ptr -> left != nullptr)**

**{**

**ptr = ptr -> left;**

**} else**

**{**

**add\_Node(ptr, a, false);**

**break;**

**}**

**}**

**if (a == ptr -> n) {break;}**

**}**

**ptr = first;**

**}**

**ptr = first;**

**int to\_find;**

**wcout<<L"Введите число, которое нужно найти: ";**

**cin >> to\_find;**

**find(ptr, to\_find);**

**int choice =0;**

**wcout << L"Выберитеспособвывода:" << endl << L"1 - Инфиксный" << endl << L"2 - Постфиксный" << endl << L"3 - Префиксный" << endl;**

**cin >> choice;**

**ptr = first;**

**while ((choice != 1)&&(choice != 2)&&(choice != 3))**

**{**

**wcout<<L"Введите корректное число:";**

**cin >> choice;**

**}**

**if (choice == 1)**

**{**

**infix(ptr);**

**}**

**if (choice == 2)**

**{**

**postfix(ptr);**

**}**

**if (choice == 3)**

**{**

**prefix(ptr);**

**}**

**ptr = first;**

**int height = tree\_height(ptr);**

**wcout<<L"Вывод дерева:" <<endl;**

**ptr = first;**

**for (int i = 1; i <= height; i++)**

**{**

**wcout << L"Уровень " << i << ":" << endl;**

**vyvod(ptr, i);**

**}**

**delete\_tree(first);**

**return 0;**

**}**