МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

НИЖЕГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ

УНИВЕРСИТЕТ им. Р.Е.АЛЕКСЕЕВА

Институт радиоэлектроники и информационных технологий

Кафедра прикладной математики

Рекурсия. Бинарные деревья

ОТЧЕТ

По лабораторной работе

по дисциплине

Технология программирования

РУКОВОДИТЕЛЬ:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Санников А.Н.

СТУДЕНТ:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Зырянов Е.А.

22-ПМ-1

Работа защищена «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_

С оценкой \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Задание на лабораторную работу:

1. Дано бинарное дерево. Найти ветви с мах числом ветвлений.

Цель работы:

Познакомиться с рекурсией и бинарными деревьями.

Ход работы:

1) Написали программу:

#include <iostream>

struct NodeList { //структура элемента

private: //

int data; //

NodeList\* next; //

public: //

NodeList() { //

next = NULL; //

} //

int get\_data() { //

return data; //

} //

NodeList\* get\_next() { //

return next; //

} //

void set\_data(int data) { //

this->data = data; //

} //

void set\_next(NodeList\* next) { //

this->next = next; //

} //

int\* get\_data\_link() { //

return &data; //

} //

};

class list { //список

private: //

NodeList\* head = NULL;

public:

void push\_back(int data) { //вставить в конец

if (head == NULL) { //

head = new NodeList; //

head->set\_data(data); //

return; //

} //

NodeList\* tmp = head; //

while (tmp->get\_next()) { //

tmp = tmp->get\_next(); //

} //

tmp->set\_next(new NodeList); //

tmp->get\_next()->set\_data(data); //

}

void print() { //вывести

NodeList\* tmp = head; //

while (tmp) { //

std::cout << tmp->get\_data() << ' '; //

tmp = tmp->get\_next(); //

} //

std::cout << '\n'; //

}

int len() {

NodeList\* tmp = head;

int c = 0;

while (tmp) { //

c += 1; //

tmp = tmp->get\_next(); //

}

return c;

}

int\* operator[](int i) { //перегрузка оператора

NodeList\* tmp = head; //

for (int j = 0; j < i; j++) { //

tmp = tmp->get\_next(); //

} //

return tmp->get\_data\_link(); //

}

};

class BinaryTree {

public:

struct NodeTree {

int value;

int count;

NodeTree\* left;

NodeTree\* right;

};

private:

NodeTree\* root;

NodeTree\* AddNode(NodeTree\* root, int value) {

if (root == nullptr) {

root = new NodeTree;

root->value = value;

root->count = 1;

root->left = root->right = nullptr;

}

else if (root->value > value) {

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

}

else if (root->value < value) {

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

}

else {

root->count++;

}

return root;

}

void PrintTree(NodeTree\* root) {

if (root == nullptr)

return;

PrintTree(root->left);

for (int i = 0; i < root->count; ++i)

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

PrintTree(root->right);

}

void DeleteTree(NodeTree\* root) {

if (root == nullptr)

return;

else {

DeleteTree(root->left);

DeleteTree(root->right);

delete root;

}

}

int maxDepth(NodeTree\* node) {

if (node == nullptr)

return 0;

int left = maxDepth(node->left);

int right = maxDepth(node->right);

return std::max(left, right) + 1;

}

void printMaxBranches(NodeTree\* Node,list path, int pathLen) {

if (Node == nullptr)

return;

if (path.len() <= pathLen)

path.push\_back(Node->value);

else

\*(path[pathLen]) = Node->value;

pathLen++;

if (Node->left == nullptr && Node->right == nullptr && pathLen == maxDepth(root))

path.print();

else {

printMaxBranches(Node->left, path, pathLen);

printMaxBranches(Node->right, path, pathLen);

}

}

public:

BinaryTree() : root(nullptr) {}

~BinaryTree() {

if (root)

DeleteTree(root);

}

void Add(int value) {

root = AddNode(root, value);

}

void Print() {

PrintTree(root);

}

void Clear() {

if (root) {

DeleteTree(root);

root = nullptr;

}

}

void PrintMaxBranch() {

int max\_branches = maxDepth(root);

std::cout << "Максимальное число ветвлений: " << max\_branches << std::endl;

}

void PrintMaxBranches() {

list path;

std::cout << "Максимальные ветки: " << '\n';

printMaxBranches(root, path, 0);

}

};

int main() {

using namespace std;

setlocale(LC\_ALL, "Russian");

BinaryTree Tree;

int a,n;

cout << "Введите количество узлов: ";

cin >> n; cout << '\n';

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

cin >> a;

Tree.Add(a);

}

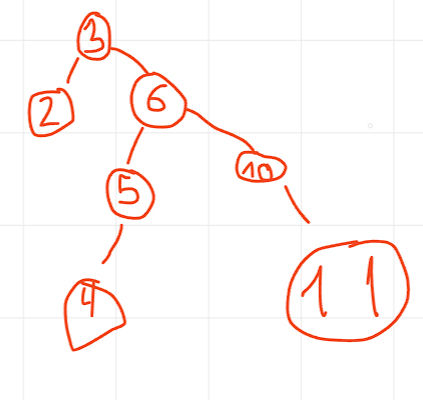
Tree.PrintMaxBranch();

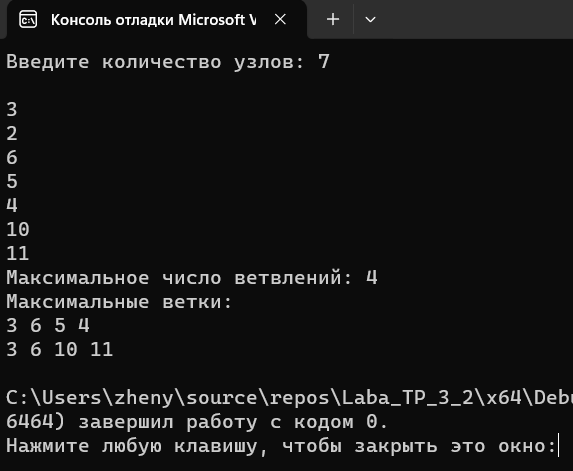
Tree.PrintMaxBranches();

return 0;

}

Результат:





Вывод: Познакомились с рекурсией и бинарными деревьями.