ГУО «БГУИР»

Факультет информационных технологий и управления

Кафедра вычислительных методов и программирования

Отчет по

Лабораторной работе №6

ПРОГРАММИРОВАНИЕ С ИСПОЛЬЗОВАНИЕМ ОДНОМЕРНЫХ МАССИВОВ

Подготовил:

Студент гр.321702

Казаченко Вадим

Проверила:

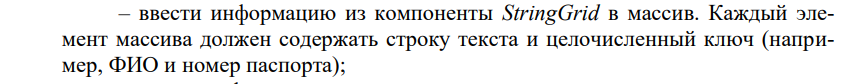
Семижон Е.А.

Минск 2023

Цель: изучить алгоритмы обработки данных с использованием нелинейных структур в виде дерева.

Вариант №10*.*

Написать программу, выполняющую алгоритм, согласно заданию.



*Код:*

#include <iostream>

#include<string>

using namespace std;

struct Tree {

string text;

int key;

Tree\* left;

Tree\* right;

};

void ADD(Tree\*& root, string text, int key) {

if (root == nullptr) {

root = new Tree;

root->text = text;

root->key = key;

root->left = nullptr;

root->right = nullptr;

}

else {

if (key < root->key) {

ADD(root->left, text, key);

}

else {

ADD(root->right, text, key);

}

}

}

void Pramoi(Tree\* root) { // Прямой обход(Root-Left-Right)

if (root == nullptr) {

return;

}

cout << "Текст: " << root->text << ", Ключ: " << root->key << endl;

Pramoi(root->left);

Pramoi(root->right);

}

void Obratny(Tree\* root) { // Обратный обход(Left-Right-Root)

if (root == nullptr) {

return;

}

Obratny(root->left);

Obratny(root->right);

cout << "Текст: " << root->text << ", Ключ: " << root->key << endl;

}

void Vozrast(Tree\* root) { // Центрированный обход(Left-Root-Right)

if (root == nullptr) {

return;

}

Vozrast(root->left);

cout << "Текст: " << root->text << ", Ключ: " << root->key << endl;

Vozrast(root->right);

}

Tree\* min(Tree\* node) {

while (node->left != nullptr)

node = node->left;

return node;

}

Tree\* dell(Tree\* root, int key) {

if (root == nullptr) {

return root;

}

if (key < root->key) {

root->left = dell(root->left, key);

}

else if (key > root->key) {

root->right = dell(root->right, key);

}

else {

if (root->left != nullptr) {

Tree\* temp = root->right;

delete root;

return temp;

}

else if (root->right != nullptr) {

Tree\* temp = root->right;

delete root;

return temp;

}

Tree\* temp = min(root->right);

root->key = temp->key;

root->text = temp->text;

root->right = dell(root->right, temp->key);

}

return root;

}

int count\_nodes\_with\_two\_children(Tree\* root) {

if (root == nullptr) {

return 0;

}

int count = 0;

if (root->left != nullptr && root->right != nullptr) {

count++;

}

count += count\_nodes\_with\_two\_children(root->left);

count += count\_nodes\_with\_two\_children(root->right);

return count;

}

void deleteTree(Tree\* root) {

if (root == nullptr) {

return;

}

deleteTree(root->left);

deleteTree(root->right);

delete root;

}

int main() {

setlocale(LC\_ALL, "RU");

Tree\* root = nullptr;

int choice;

do {

string text;

int key;

cout << "---Новая запись---" << endl;

cout << "Введите имя: ";

getline(cin, text);

cout << "Введите номер паспорта: ";

cin >> key;

cin.ignore();

ADD(root, text, key);

cout << "Хотите добавить еще запись? (1 - да)(2 - нет): ";

cin >> choice;

cin.ignore();

} while (choice == 1);

cout << "Прямой обход дерева:" << endl;

Pramoi(root);

cout << "\nОбратный обход дерева:" << endl;

Obratny(root);

cout << "\nДерево в порядке возрастания ключа:" << endl;

Vozrast(root);

int nodes\_with\_two\_children = count\_nodes\_with\_two\_children(root);

cout << "Число узлов с двумя дочерними узлами: " << nodes\_with\_two\_children << endl;

int key;

cout << "Введите ключ для удаления записи: ";

cin >> key;

dell(root, key);

cout << "Прямой обход дерева:" << endl;

Pramoi(root);

cout << "\nОбратный обход дерева:" << endl;

Obratny(root);

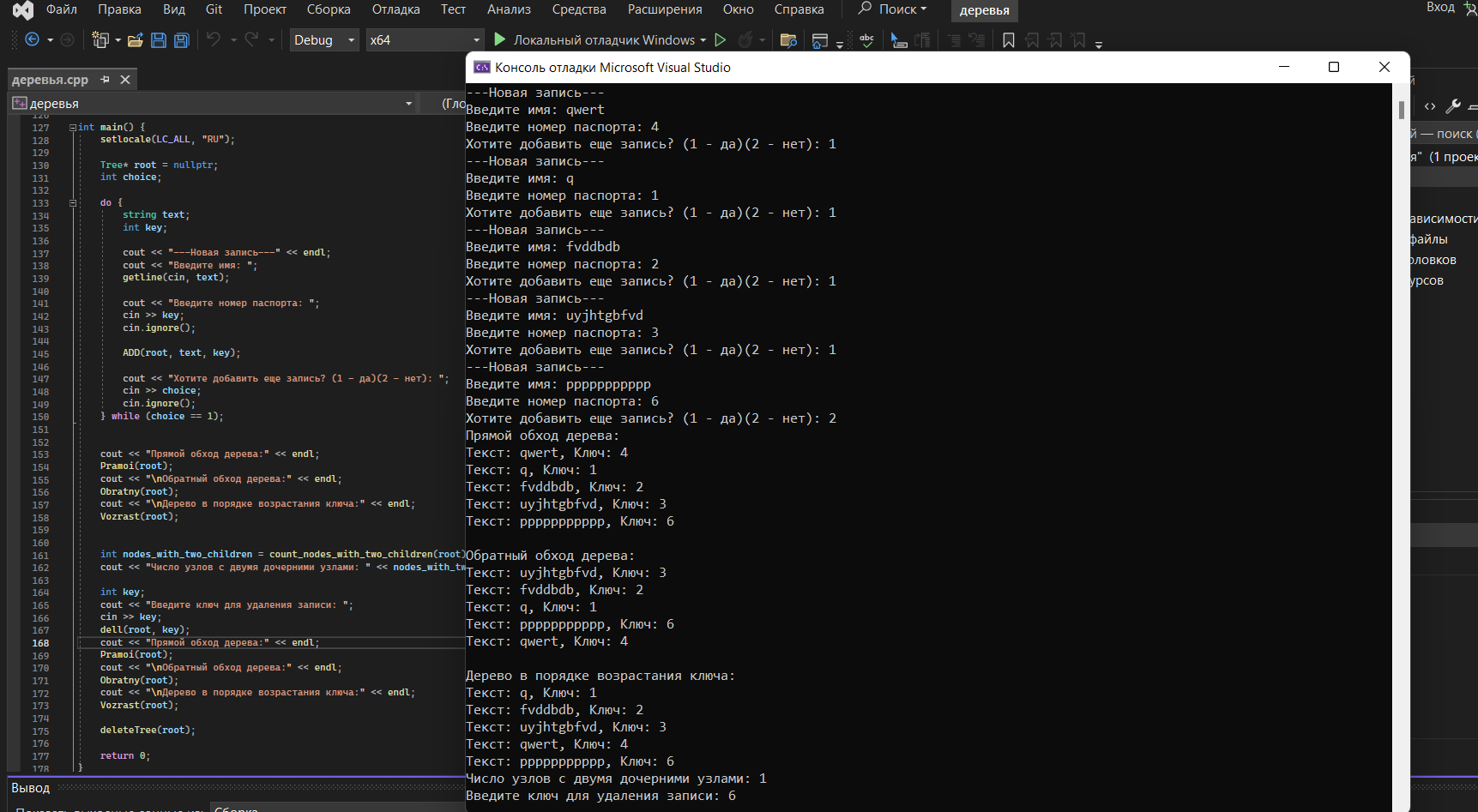
cout << "\nДерево в порядке возрастания ключа:" << endl;

Vozrast(root);

deleteTree(root);

return 0;

}



Вывод: создал программу, которая работает с деревом и определяет число узлов в дереве у которых есть две дочери.