Министерство образования Республики Беларусь

Учреждение образования

«Брестский Государственный технический университет»

Кафедра ИИТ

**Лабораторная работа №14**

По дисциплине «Основы алгоритмизации и программирования»

Тема: «Динамические структуры: списки и деревья»

**Выполнил:**

Студент 1 курса

Группы ИИ-21

Карагодин Д.Л.

**Проверил:**

Гирель Т.Н

Брест 2022

**Цель работы:** Приобретение навыков работы с динамической памятью и указателями на С/C++. Изучение принципов работы с динамическими структурами данных: списками и деревьями.

**Вариант №1**

**Текст задания**

**Написать программу, которая вводит с клавиатуры список целых чисел, считает длину списка, затем, из элементов списка создает сбалансированное дерево и подсчитывает среднее арифметическое элементов дерева.**

**Текст программы**

#include <iostream>

using namespace std;

struct Node {

int key;

Node\* left;

Node\* right;

int height;

};

short counts = 0, summ = 0;

int height(Node\* N) {

if (N == NULL) return 0;

return N->height;}

int max(int a, int b) {return (a > b) ? a : b;}

Node\* newNode(int key) {

counts++;

summ += key;

Node\* node = new Node();

node->key = key;

node->left = NULL;

node->right = NULL;

node->height = 1;

return (node);}

Node\* rightRotate(Node\* y) {

Node\* x = y->left;

Node\* T2 = x->right;

x->right = y;

y->left = T2;

y->height = max(height(y->left),

height(y->right)) +

1;

x->height = max(height(x->left),

height(x->right)) +

1;

return x;}

Node\* leftRotate(Node\* x) {

Node\* y = x->right;

Node\* T2 = y->left;

y->left = x;

x->right = T2;

x->height = max(height(x->left),

height(x->right)) +

1;

y->height = max(height(y->left),

height(y->right)) +

1;

return y;}

int getBalanceFactor(Node\* N) {

if (N == NULL) return 0;

return height(N->left) -

height(N->right);}

Node\* insertNode(Node\* node, int key) {

if (node == NULL)

return (newNode(key));

if (key < node->key)

node->left = insertNode(node->left, key);

else if (key > node->key)

node->right = insertNode(node->right, key);

else return node;

node->height = 1 + max(height(node->left),

height(node->right));

int balanceFactor = getBalanceFactor(node);

if (balanceFactor > 1) {

if (key < node->left->key) {

return rightRotate(node);}

else if (key > node->left->key) {

node->left = leftRotate(node->left);

return rightRotate(node);}}

if (balanceFactor < -1) {

if (key > node->right->key) {

return leftRotate(node); }

else if (key < node->right->key) {

node->right = rightRotate(node->right);

return leftRotate(node);}}

return node;}

void printTree(Node\* root, string indent, bool last) {

if (root != nullptr) {

cout << indent;

if (last) {

cout << "R----";

indent += " ";}

else {

cout << "L----";

indent += "| ";}

cout << root->key << endl;

printTree(root->left, indent, false);

printTree(root->right, indent, true);}}

int main() {

Node\* root = NULL;

short exit = 1;

while (true) {

cout << "Вводите значения элемента дерева(0 для отсановки): ";

cin >> exit;

if (exit == 0) break;

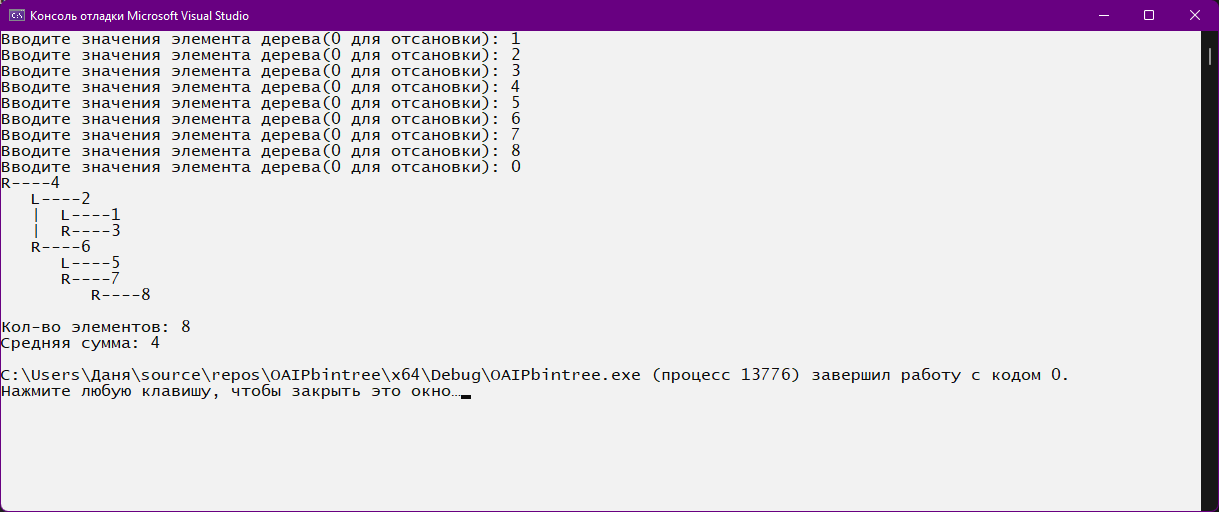
root = insertNode(root, exit);

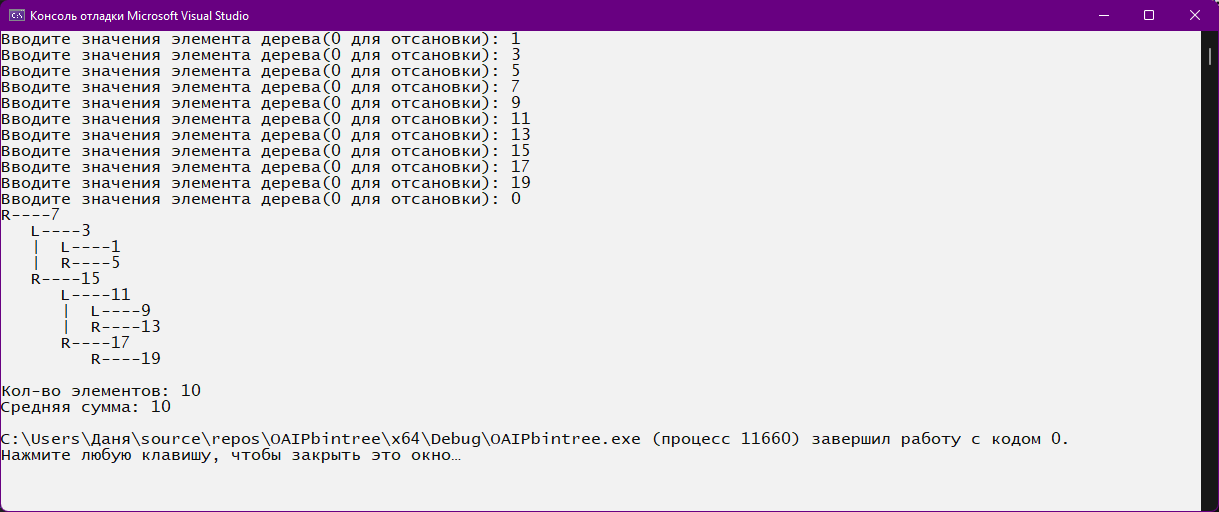
}

printTree(root, "", 1);

cout << "\nКол-во элементов: " << counts <<endl<<"Средняя сумма: "<<summ/counts<<endl;}

**Результат работы**





**Вывод:** Приобрёл навыки работы с динамической памятью и указателями на С/C++. Изучил принципы работы с динамическими структурами данных: списками и деревьями.