**Лабораторная работа №2**

**Бинарные деревья**

**ВАРИАНТ 3**

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.

ЗАДАНИЕ

Написать рекурсивную функцию, которая находит наименьший элемент дерева.

ПРОГРАММА

#include <iostream>

#include <time.h>

using namespace std;

class Tree {

private:

struct Node {

int data;

Node\* left, \* right;

};

Node\* root;

int minim = 9999999;

void show\_tree\_1(Node\* node) {

if (node != NULL) {

cout << node->data << " ";

show\_tree\_1(node->left);

show\_tree\_1(node->right);

}

}

void show\_tree\_2(Node\* node) {

if (node != NULL) {

show\_tree\_2(node->left);

cout << node->data << " ";

show\_tree\_2(node->right);

}

}

void show\_tree\_3(Node\* node) {

if (node != NULL) {

show\_tree\_3(node->left);

show\_tree\_3(node->right);

cout << node->data << " ";

}

}

void minimum(Node\* node) {

if (node != NULL) {

if (node->data < minim) minim = node->data;

minimum(node->left);

minimum(node->right);

}

}

public:

Tree() {

root = NULL;

}

void add(int value) {

if (root == NULL) {

root = new Node;

root->data = value;

root->left = NULL; root->right = NULL;

}

else {

Node\* tmp = root;

while (root->left != NULL && root->right != NULL) {

if (rand() % 2 == 0) root = root->left;

else root = root->right;

}

if (root->left == NULL) {

root->left = new Node;

root = root->left;

root->data = value;

root->left = NULL; root->right = NULL;

root = tmp;

}

else {

root->right = new Node;

root = root->right;

root->data = value;

root->left = NULL; root->right = NULL;

root = tmp;

}

}

}

void show1() {

show\_tree\_1(root);

cout << endl;

}

void min1() {

minimum(root);

cout << "Min: " << minim << endl;

}

void show2() {

show\_tree\_2(root);

cout << endl;

}

void show3() {

show\_tree\_3(root);

cout << endl;

}

};

int main() {

srand(time(0));

Tree tr;

int size\_tr = rand() % 51;

cout << "Size Tree: " << size\_tr << endl;

for (int i = 0; i < size\_tr; i++) tr.add(rand() % 101);

cout << endl;

if (size\_tr == 0) cout << "Tree is empty" << endl;

else {

cout << "Direct Bypass" << endl;

tr.show1();

cout << endl;

cout << endl;

cout << "Reverse bypass" << endl;

tr.show2();

cout << endl;

cout << endl;

cout << "Symmetric Bypass" << endl;

tr.show3();

cout << endl;

cout << endl;

tr.min1();

}

system("pause");

return 0;

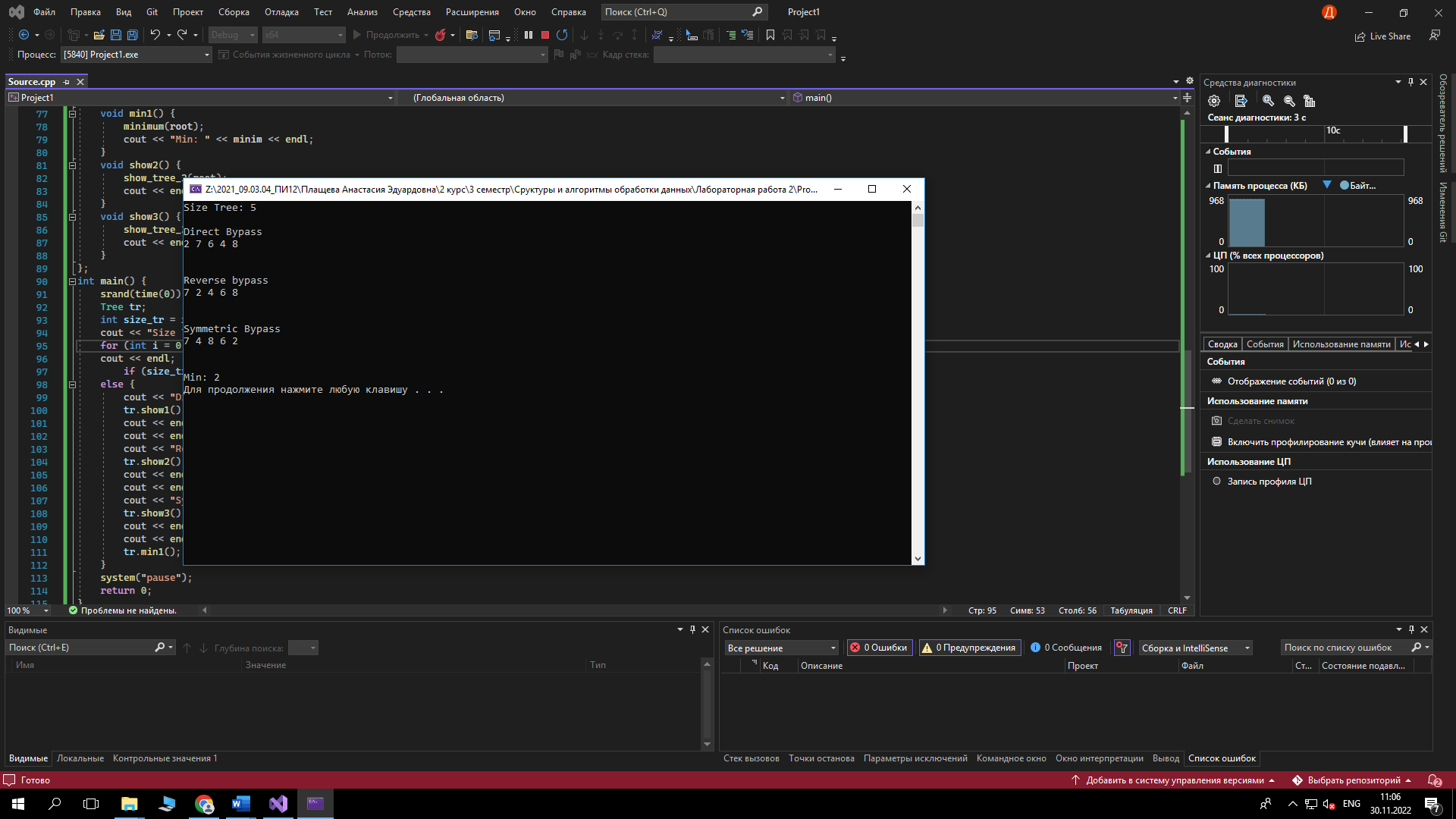
}

АЛГОРИТМ

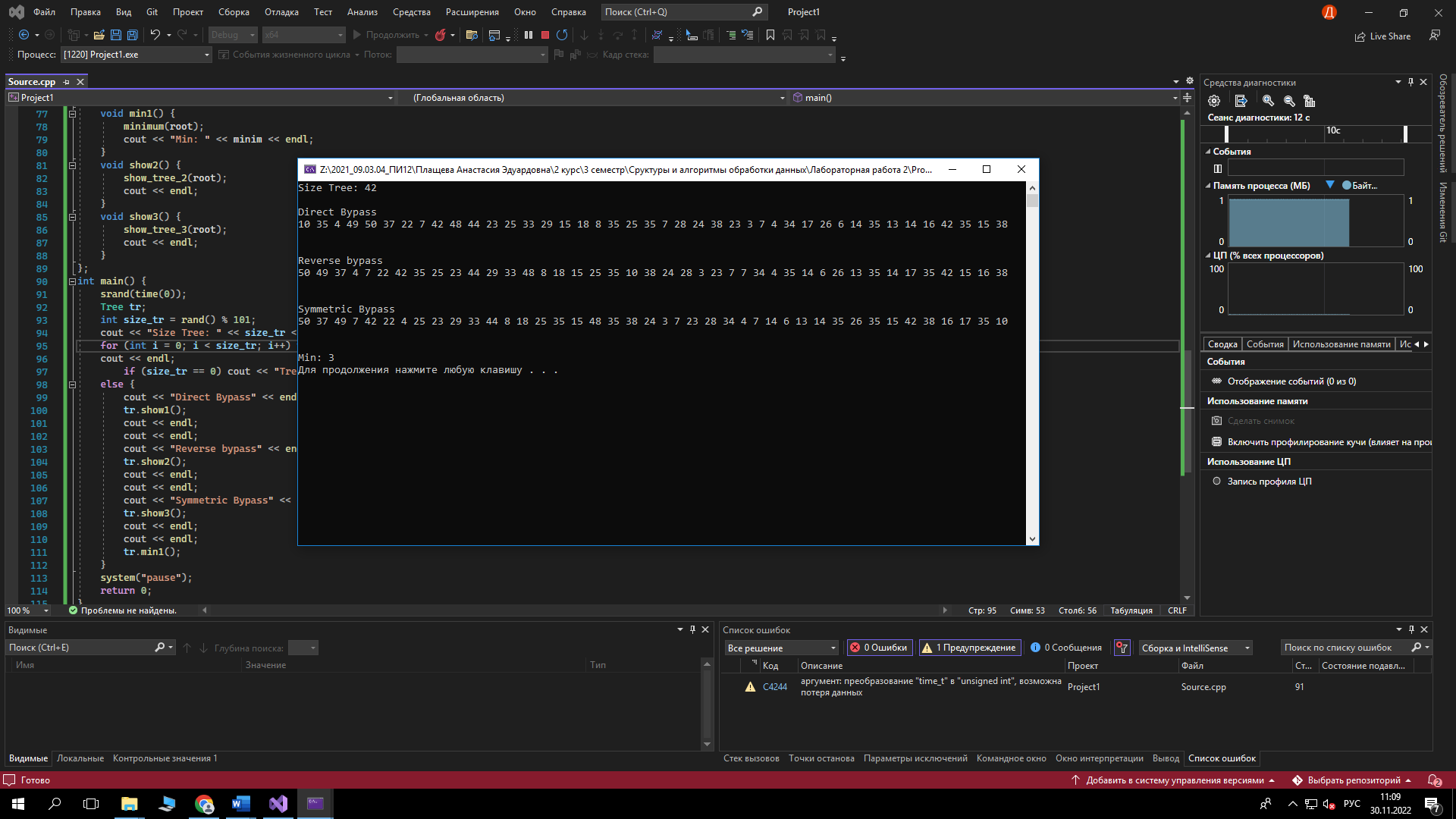
1. Объявляем класс дерева Tree;

ТЕСТ

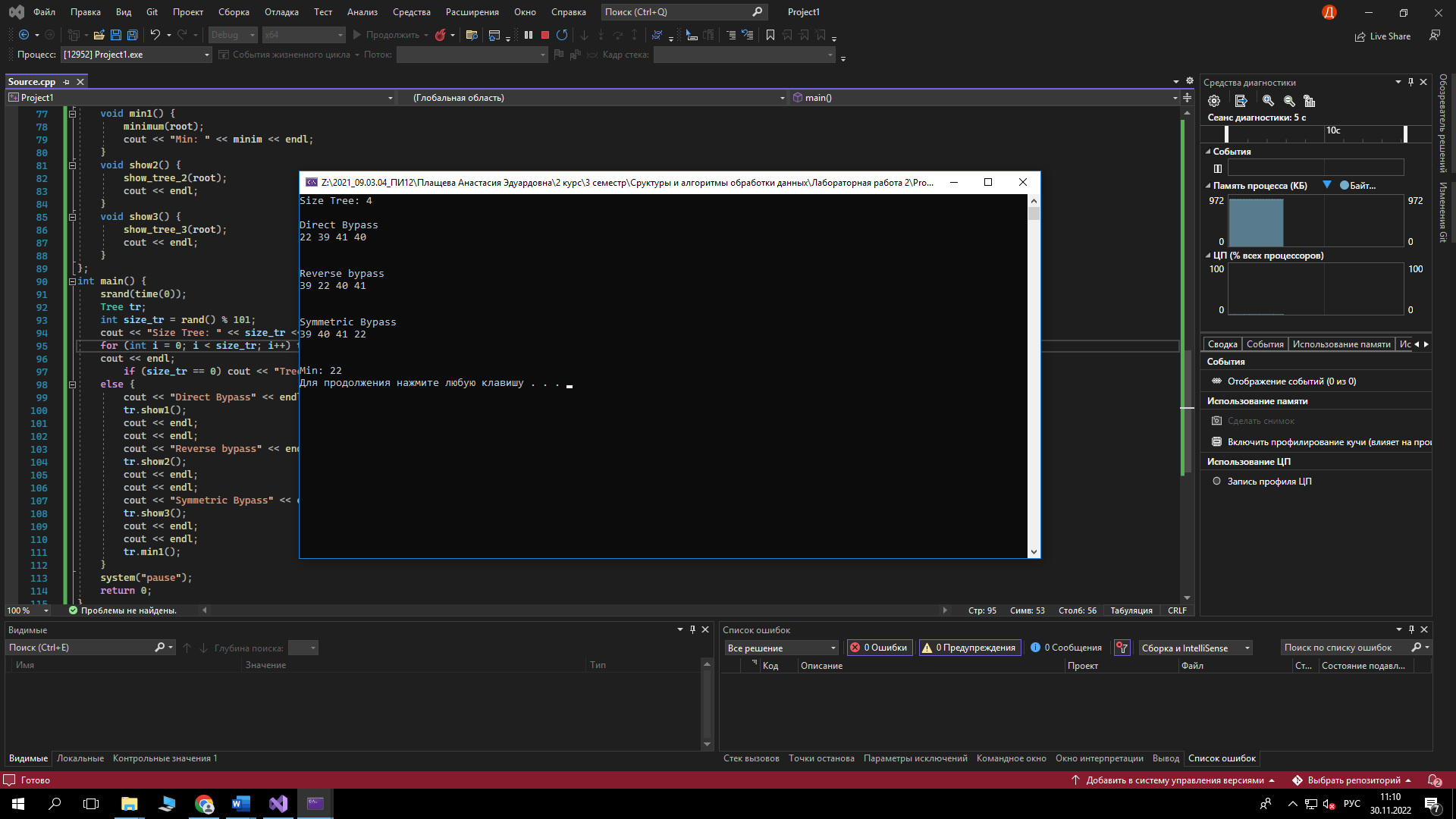
Тест 1. Наименьшее значение дерева равно 2.



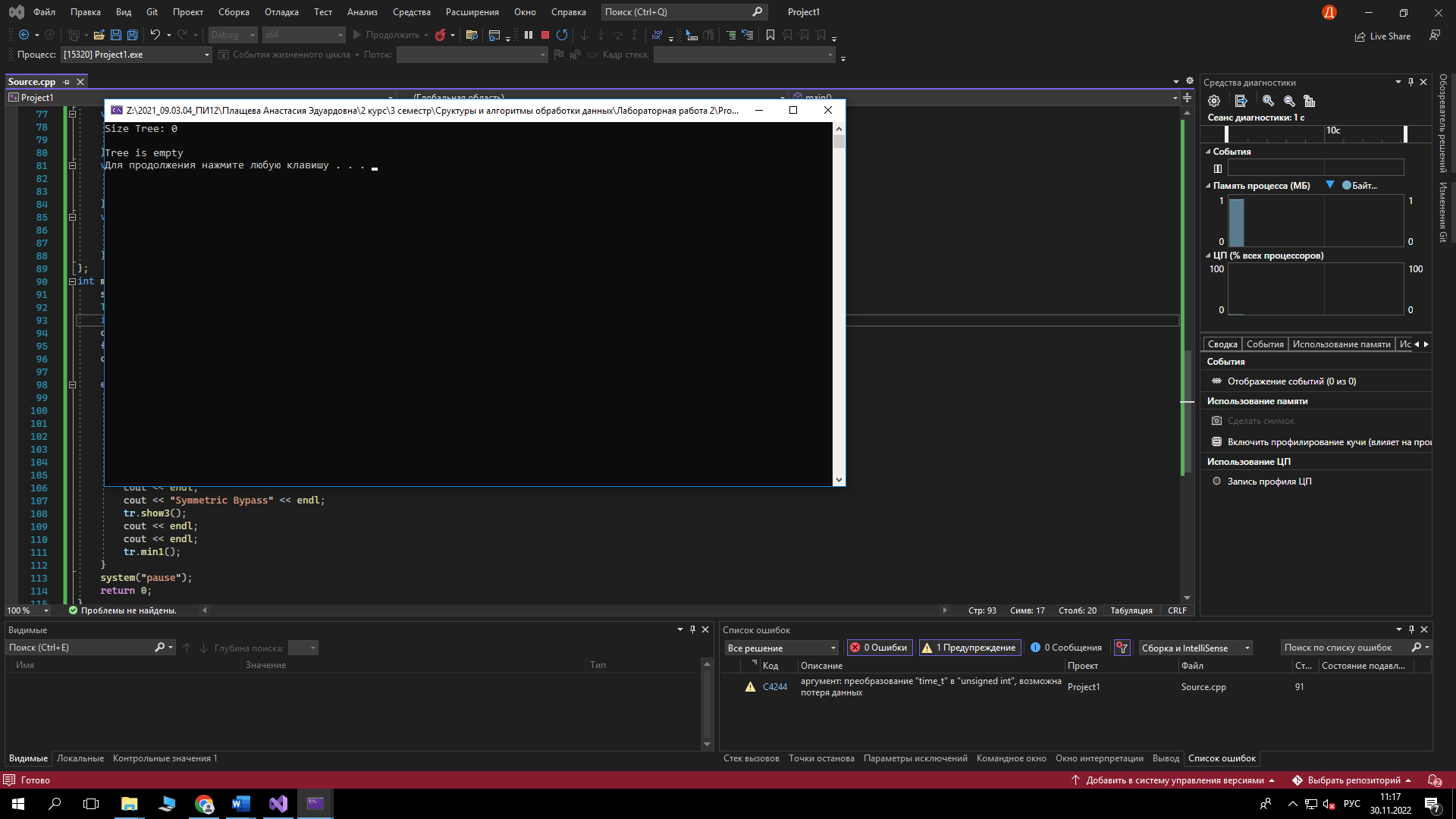
Тест 2. Наименьшее значение дерева равно 3.



Тест 3. Наименьшее значение дерева равно 22.



Тест 4. Дерево пустое



ВЫВОД

Выполняя задание, был закреплён материал по созданию и работе с бинарными деревьями.