**Лабораторная работа №2**

**Бинарные деревья**

**10 ВАРИАНТ**

Задание

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. Выполнить следующее задание: “**Написать рекурсивную функцию, которая считает количество листов дерева**”.

Программный код

#include<iostream>

using namespace std;

//В этой структуре определены три поля : data для хранения значения узла и указатели left и right для связи с левым и правым потомком.

struct Node

{

int data;

Node\* left, \* right;

};

// Функция add отвечает за добавление нового узла в бинарное дерево.

// Если переданный узел node равен NULL, создается новый узел с данными x и указателями на потомков,

// иначе функция случайным образом выбирает, добавлять новый узел в левую или правую ветвь, и рекурсивно вызывает себя для соответствующего поддерева.

void add(int x, Node\*& node)

{

if (node == NULL)

{

node = new Node;

node->data = x;

node->left = NULL;

node->right = NULL;

}

else

{

int rand\_branch = rand() % 2;

if (rand\_branch) add(x, node->right);

else add(x, node->left);

}

}

// В данной функции генерируется случайное бинарное дерево заданного размера size.

// Для каждого узла в цикле генерируется случайное значение randomValue от 0 до 199, и затем это значение добавляется в дерево с помощью вызова функции add.

void buildRandomTree(Node\*& root, int size)

{

srand(time(0));

for (int i = 0; i < size; i++)

{

int randomValue = rand() % 200; // Генерация случайного числа от 0 до 99

add(randomValue, root);

}

}

// Прямой обход:

// Функция straightPrintTree выводит значения узлов дерева в порядке прямого обхода:

// сначала текущий узел, затем левое поддерево и правое поддерево.

// Рекурсивно обрабатываются левое и правое поддеревья.

void straightPrintTree(Node\* node)

{

if (node != NULL)

{

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

straightPrintTree(node->left);

straightPrintTree(node->right);

}

}

// Обратный обход:

// В функции reversePrintTree значения узлов выводятся в обратном порядке обхода:

// сначала левое поддерево, затем правое поддерево и текущий узел.

// Рекурсивно обрабатываются левое и правое поддеревья.

void reversePrintTree(Node\* node)

{

if(node!=0)

{

reversePrintTree(node->left);

reversePrintTree(node->right);

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

}

}

// Cимметричный обход:

// Функция symmetricalPrintTree осуществляет симметричный обход дерева, выводя значения узлов в порядке:

// левое поддерево, текущий узел, правое поддерево.

// Рекурсивно обрабатываются левое и правое поддеревья.

void symmetricalPrintTree(Node\* node)

{

if (node != 0)

{

symmetricalPrintTree(node->left);

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

symmetricalPrintTree(node->right);

}

}

// Количество листьев в дереве:

// Функция countingLeaves подсчитывает количество листьев в бинарном дереве.

// Если узел равен NULL, возвращается 0.

// Если узел является листовым (не имеет потомков), возвращается 1.

// Иначе рекурсивно вызывается для левого и правого поддерева, и возвращается их сумма.

int countingLeaves(Node\* node)

{

if (node == NULL) return 0;

if (node->left == NULL && node->right == NULL) return 1;

return countingLeaves(node->left) + countingLeaves(node->right);

}

int main()

{

setlocale(LC\_ALL, "ru");

srand(time(0));

Node\* root = NULL;

int elements=rand()%1+10;

buildRandomTree(root, elements);

cout << "Прямой обход: ";

straightPrintTree(root);

cout << endl;

cout << "Обратный обход: ";

reversePrintTree(root);

cout << endl;

cout << "Симметричный обход: ";

symmetricalPrintTree(root);

cout << endl;

int leaves = countingLeaves(root);

cout << "Количество листьев в дереве: " << leaves << endl;

return 0;

}

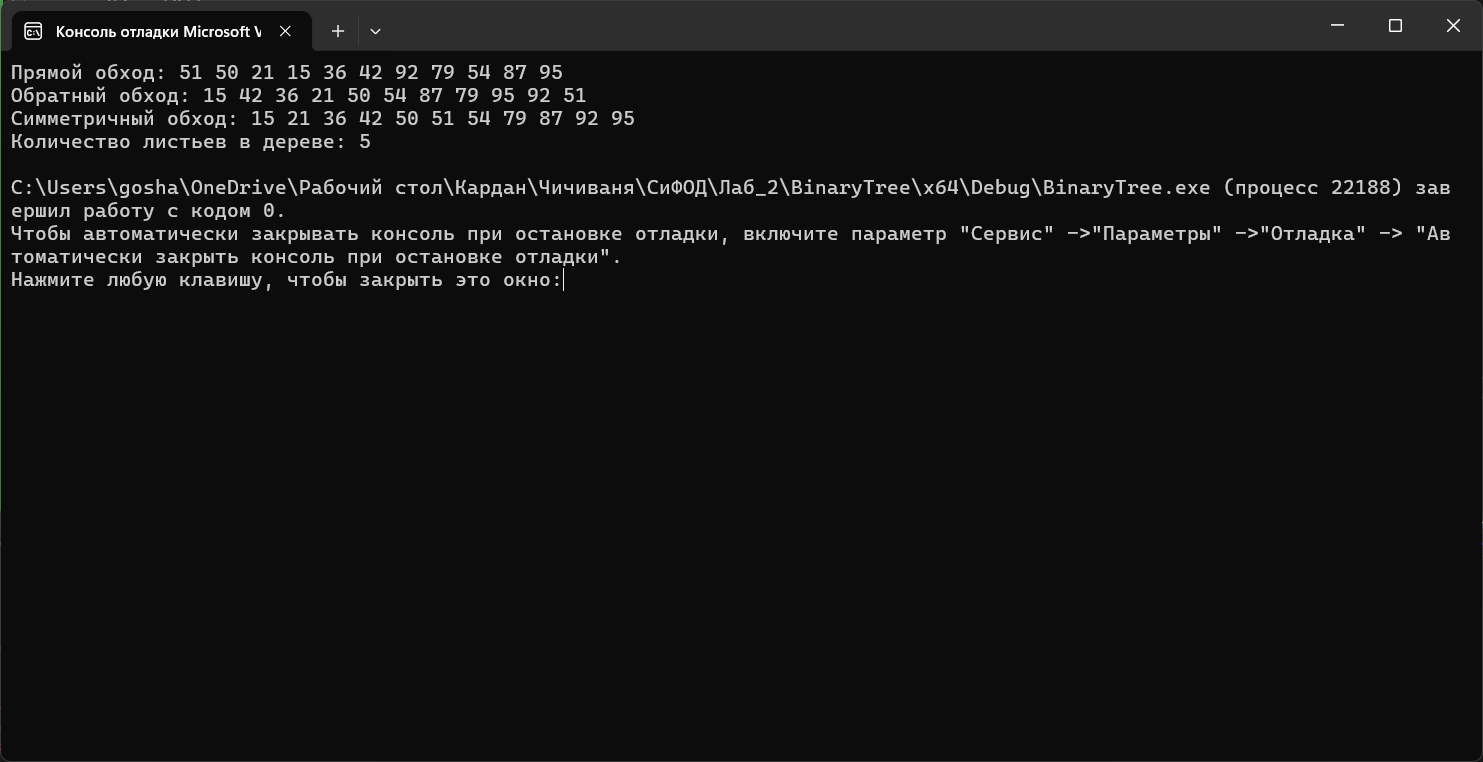


Рис. 1. Вывод программы

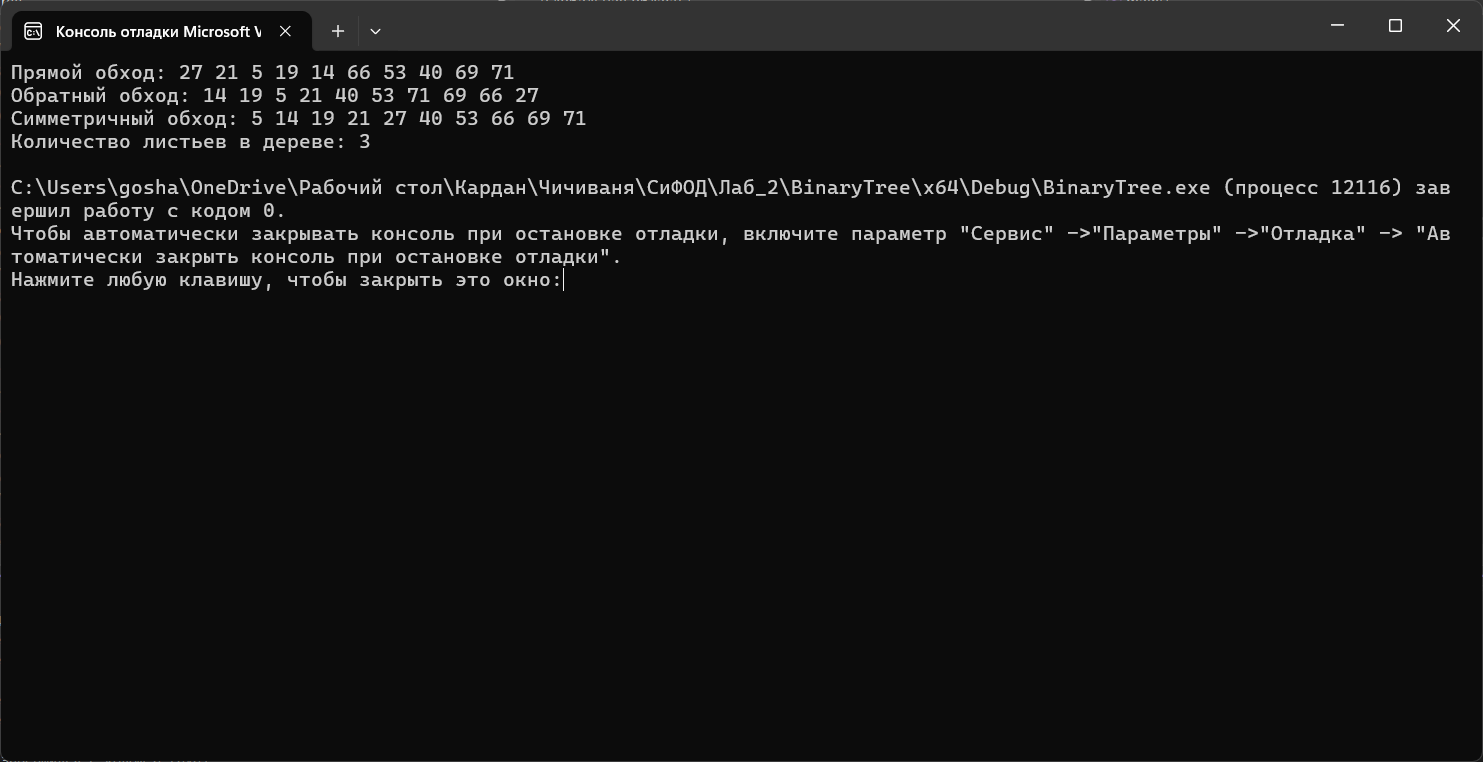


Рис. 2. Вывод программы

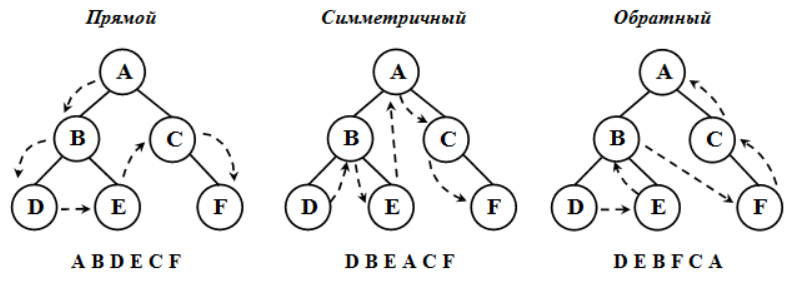


Рис. 3. Обходы деревьев в глубину

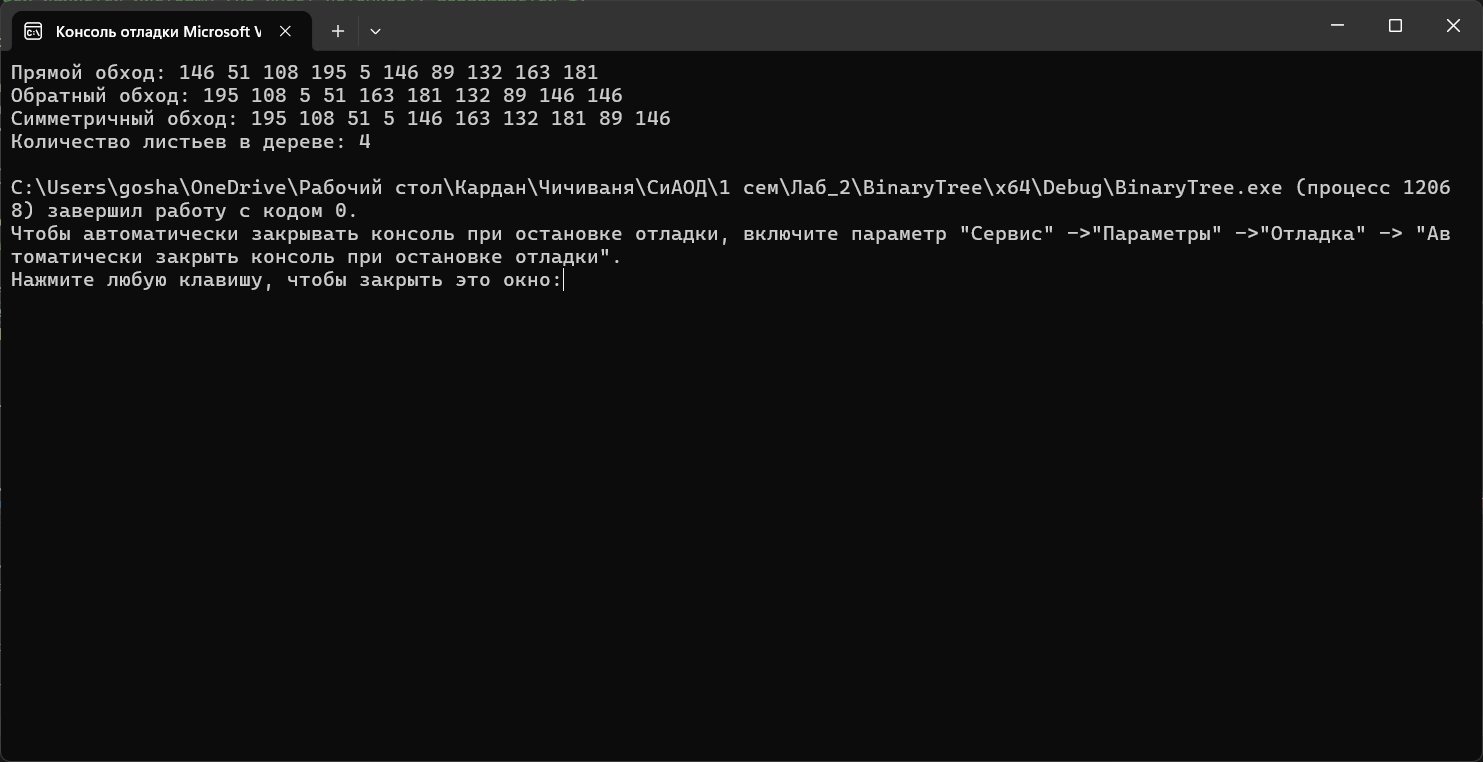


Рис. 4. Вывод программы

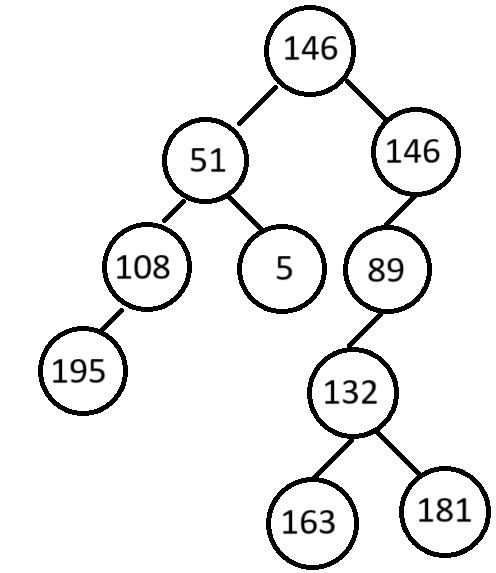


Рис. 5. Бинарное дерево по выводу программы

Описание алгоритма работы программы

1. Структура Node:

- Определяется структура Node, которая содержит целочисленное значение data и указатели на левого и правого потомков узла. Эта структура используется для представления каждого узла в бинарном дереве.

2. Функция add(int x, Node\*&node):

- Если переданный узел node равен NULL, то создается новый узел с значением x и указателями на потомков, и узел node указывает на этот новый узел.

- Иначе, если значение x меньше значения текущего узла, функция вызывается рекурсивно для левого потомка, иначе для правого потомка.

3. Функция buildRandomTree(Node\*& root, int size):

- Генерируется случайное значение для каждого узла от 0 до 99 и вызывается функция add, чтобы добавить это значение в дерево.

- В итоге, строится случайное бинарное дерево заданного размера size, используя случайно сгенерированные значения.

4. Три функции обхода дерева:

- Прямой обход (straightPrintTree):

- Выводит значение текущего узла, затем вызывает себя рекурсивно для левого поддерева, затем для правого.

- Обратный обход (reversePrintTree):

- Вызывает себя рекурсивно для левого поддерева, затем для правого, и после этого выводит значение текущего узла.

- Симметричный обход (symmetricalPrintTree):

- Вызывает себя рекурсивно для левого поддерева, затем выводит значение текущего узла, затем вызывается для правого поддерева.

5. Функция countingLeaves(Node\* node):

- Рекурсивно подсчитывает количество листьев в дереве.

- Если узел node является листом (не имеет левого и правого потомков), функция возвращает 1.

- Для каждого узла считается количество листьев в левом и правом поддеревьях, и результат слагается.

6. В функции main():

- Устанавливается русская локаль и инициализируется указатель root дерева.

- Генерируется случайное количество элементов от -5 до 5 для дерева.

- Строится случайное дерево с заданным количеством элементов.

- Выводятся элементы дерева в прямом, обратном и симметричном обходах.

- Подсчитывается количество листьев в дереве и выводится на экран.