Міністерство освіти і науки України

Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського»
Факультет інформатики та обчислювальної техніки

Кафедра інформатики та програмної інженерії

Звіт

з лабораторної роботи № 6 з дисципліни
«Основи програмування-2.
Методології програмування»

«Дерева»

Варіант 10

Виконав студент ІП-11, Друзенко Олександра Юріївна
(шифр, прізвище, ім'я, по батькові)

Перевірив Вітковська Ірина Іванівна
(прізвище, ім'я, по батькові)

Київ 2022

Мета: вивчити особливості організації і обробки дерев.

Постановка задачі:

10. Побудувати дерево, елементами якого є цілі числа. Визначити кількість вершин на n-му його рівні та кількість рівнів.

Виконання мовою C++:

1)код:

lib.h:

```
#pragma once
#include <iostream>
#include <string>

int* get_array(int& count);
```

lib.cpp:

```
#include "lib.h"

int* get_array(int& count) {
    std::cout << "Введіть масив цілих чисел: " << std::endl;
    std::string str_array;
    std::getline(std::cin, str_array);
    int counter = 0;

    for (const char& c : str_array) {
        if (c == ' ') {
            counter++;
        }
    }
    counter++;
    count = counter;

    int* num_arr = new int[counter];
    int i = 0;
    int num_counter = 0;
    while (i < str_array.length()) {
        std::string curr_num;
        while (i < str_array.length() && str_array[i] != ' ') {
            curr_num.push_back(str_array[i]);
            i++;
        }
        int trueNum = std::stoi(curr_num);
        num_arr[num_counter] = trueNum;
        num_counter++;
        i++;
    }
    return num_arr;
}
```

lab_6.cpp:

```
#include <iostream>
#include <Windows.h>
#include "Tree.h"
#include "lib.h"
```

```
int Tree::k;

int main() {
	SetConsoleCP(1251);
	SetConsoleOutputCP(1251);

	int num, vert_lev = 0, level_num = 0;
	Tree tree;

	int arr_count = 0;
	int* arr = get_array(arr_count);

	tree.makeTree(arr, 0, arr_count);
	tree.print();

	std::cout << "\nобходи дерева:\n\nпрямий:\n";
	tree.TLR(tree.get_root(), 0);

	std::cout << "\nзворотній: \n";
	tree.LRT(tree.get_root(),0);

	std::cout << "\nсиметричний: \n";
	tree.LTR(tree.get_root(),0);

	tree.count_tree_level(tree.get_root(), 1, level_num);
	std::cout << "\nкількість вершин в дереві: " << arr_count;
	std::cout << "\nкількість рівнів в дереві: " << level_num;

	std::cout << "\nведіть номер рівня: "; std::cin >> num;
	tree.print_tree_level(tree.get_root(), 1, num, vert_lev);
	std::cout << "\nкількість вершин на рівні: " << vert_lev;

}
```

Node.h:

```
#pragma once
#include <string>
class Node
{
	int data;
	Node* left, * right;
public:
	Node(int value) : data(value), left(nullptr), right(nullptr){}
	void print();
	friend class Tree;
};
```

Node.cpp:

```
#include "Node.h"
#include <iostream>

void Node::print() {
	std::cout << data << "  ";
	if (left) left->print();
	if (right) right->print();
}
```

Tree.h:

```
#pragma once
```

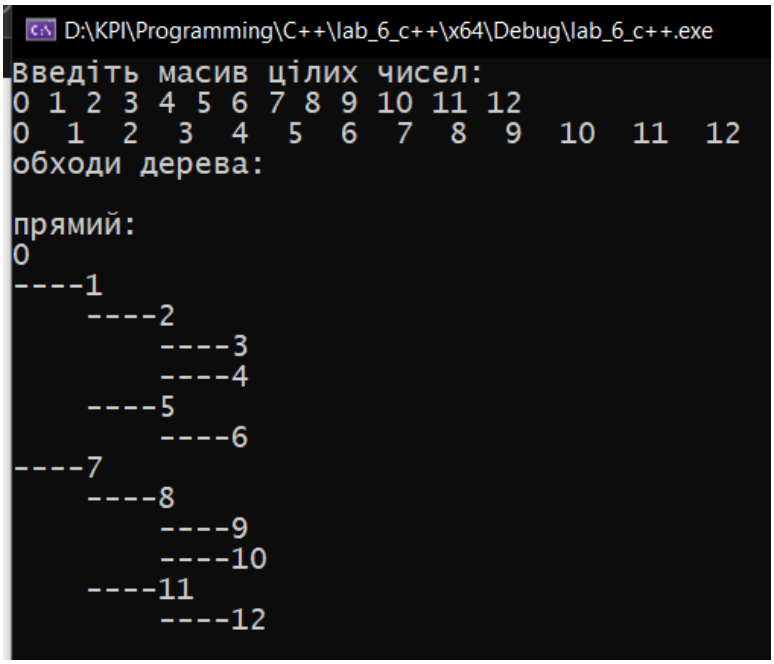
```
#include "Node.h"

class Tree
{
	Node * root;
public:
	static int k;
	Tree(): root(nullptr){}
	Node* makeTree(int[], int, int);
	Node* get_root();
	void print();
	void TLR(Node* p, int level);
	void LRT(Node* p, int level);
	void LTR(Node* p, int level);
	void count_tree_level(Node* p, int level, int& level_c);
	void print_tree_level(Node* p, int level, int level_num, int&vert_c);
};
```

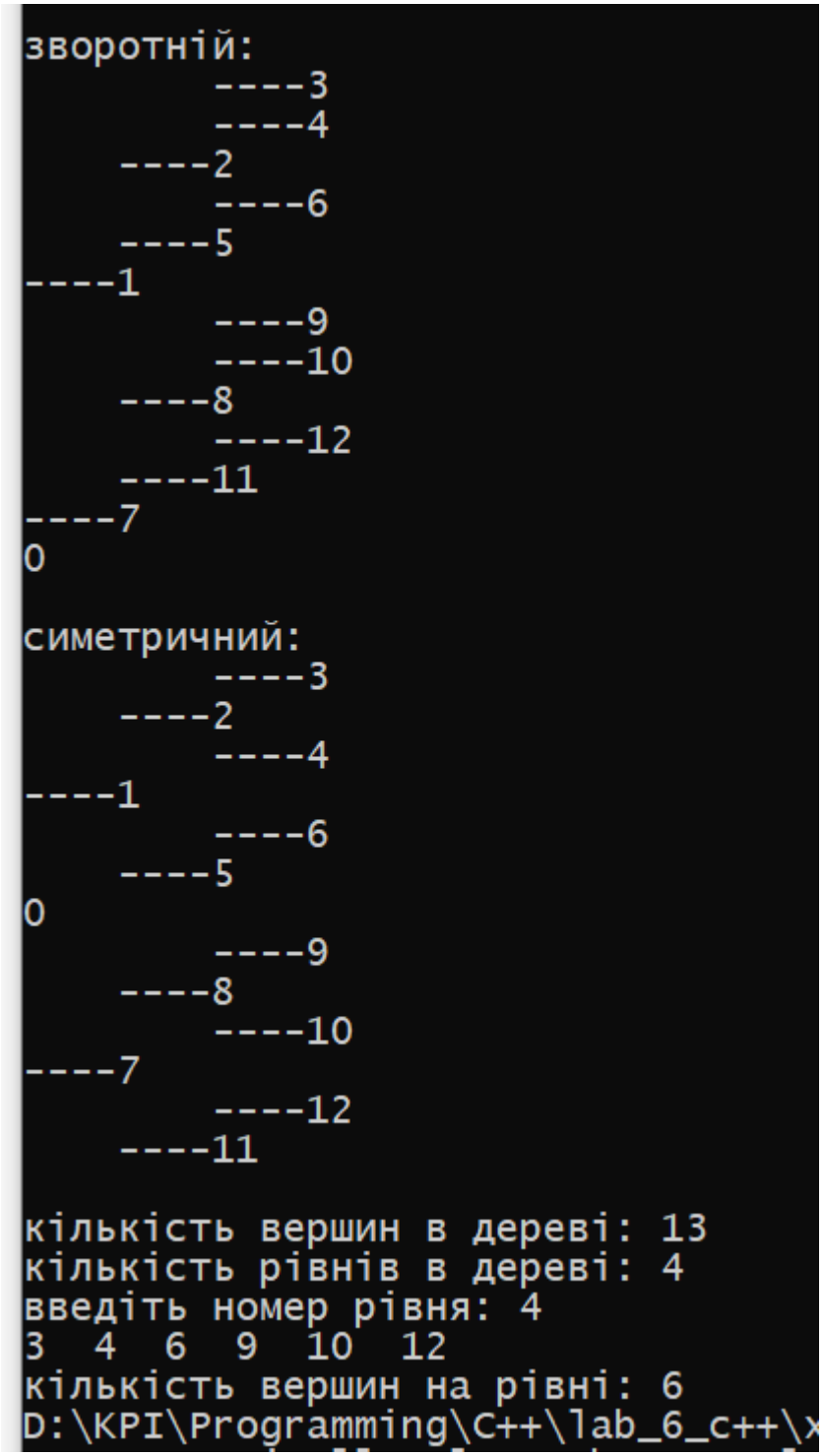
Tree.cpp:

```
#include "Tree.h"
#include <iostream>

Node* Tree::makeTree(int arr[], int from, int free) {
	if (free == 0) return nullptr;
	Node* p = new Node(arr[from]);
	if (k == 0) {
		root = p;
		k++;
	}
	int left_ch = free / 2;
	int right_ch = free - left_ch - 1;
	p->left = makeTree(arr, from + 1, left_ch);
	p->right = makeTree(arr, from + 1 + left_ch, right_ch);
	return p;
}

Node* Tree::get_root()
{
	return root;
}

void Tree::print() {
	root->print();
}

void Tree::TLR(Node* p, int level)  //прямий обхід
{
	if (p) {
		for (int i = 0; i < level-1; i++) std::cout << "    ";
		if (level != 0) std::cout << "----";
		std::cout << p->data << std::endl;
		TLR(p->left, level + 1);
		TLR(p->right, level + 1);
	}
}

void Tree::LRT(Node* p, int level)  //обхід в зворотньому порядку
{
	if (p) {

		LRT(p->left, level + 1);
		LRT(p->right, level + 1);
		for (int i = 0; i < level - 1; i++) std::cout << "    ";
```

```
            if (level != 0) std::cout << "----";
            std::cout << p->data << std::endl;
        }
}

void Tree::LTR(Node* p, int level) //симетричний обхід
{
        if (p) {

            LTR(p->left, level + 1);
            for (int i = 0; i < level - 1; i++) std::cout << "    ";
            if (level != 0) std::cout << "----";
            std::cout << p->data << std::endl;
            LTR(p->right, level + 1);
        }
}

void Tree::count_tree_level(Node* p, int level, int& level_c)
{
        if (p)
        {
            count_tree_level(p->left, level + 1, level_c);
            if (level_c < level) level_c = level;
            count_tree_level(p->right, level + 1, level_c);
        }
}

void Tree::print_tree_level(Node* p, int level, int level_num, int& vert_c)
{
        if (p)
        {
            print_tree_level(p->left, level + 1, level_num, vert_c);
            if (level == level_num) {
                std::cout << p->data << "  ";
                vert_c++;
            }
            print_tree_level(p->right, level + 1, level_num, vert_c);
        }
}
```

2)Випробування коду на C++:





Висновок. Отже, на цій лабораторній роботі я вивчила особливості організації і обробки дерев, надбала навички написання програм роботи з ними.