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. Модульне програмування»

«Дерева»

Варіант №32

Виконав студент ІП-14 Шляхтун Денис Михайлович
(шифр, прізвище, ім'я, по батькові)

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

Київ 2022

## Лабораторна робота №6

**Тема:** дерева.

**Мета**: вивчити особливості організації і обробки дерев.

## Хід роботи

**Задача**.

Побудувати дерево, елементами якого є дійсні числа. Обчислити середнє арифметичне усіх його елементів.

**Виконання задачі**.

**Випробування коду на C++.**

*Код:*

main.cpp

```
6     #include "Tree.h"
7
8     int main()
9     {
10        srand(time(NULL));
11
12        //створення дерева конструктором на замовчуванням
13        Tree* tree = new Tree();
14
15        //виведення дерева
16        treeOutput("", tree->getRoot(), false);
17
18        //виведення середнього значення вершин дерева
19        std::cout << "Average: " << tree->getAverageBFS() << std::endl;
20
21        system("pause");
22    }
```

Tree.h

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

struct Node
{
    float data;     //дані вершини
    Node* left;     //посилання на лівого нащадка
    Node* right;    //посилання на правого нащадка
public:
    //конструктор з параметром
    Node(float value);

    //геттери
    float getData() { return data; };
    Node* getLeftPointer() { return left; };
    Node* getRightPointer() { return right; };

    //вставлення нової вершини-нащадка
    //pos == 1 -> left, pos == 2 -> right
    Node* insertNodePointer(Node* value, int pos);
};
```

```
23      //введення кількості елементів дерева
24      int numInput();
25
26      //генерація масиву дійсних чисел заданого розміру
27      float* arrGen(int num);
28
29      //виведення дерева
30      void treeOutput(const std::string& prefix, Node* node, bool isLeft);
31
32      class Tree
33      {
34          Node* root; //корінь дерева
35
36          //заповнення дерева заданими значеннями масиву
37          Node* createTree(float* arr, int from, int num);
38      public:
39          //конструктор за замовчуванням
40          Tree() : Tree(numInput()) {};
41
42          //конструктор з параметром, заповнення дерева випадковими значеннями
43          Tree(int num) : Tree(arrGen(num), num) {};
44
45          //конструктор з двома параметрами заповнення дерева
46          Tree(float* arr, int num) { root = createTree(arr, 0, num); };
47
48          //геттер
49          Node* getRoot() { return root; };
50
51          //знаходження середнього значення вершин дерева
52          float getAverageBFS();
53      };
```

Tree.cpp

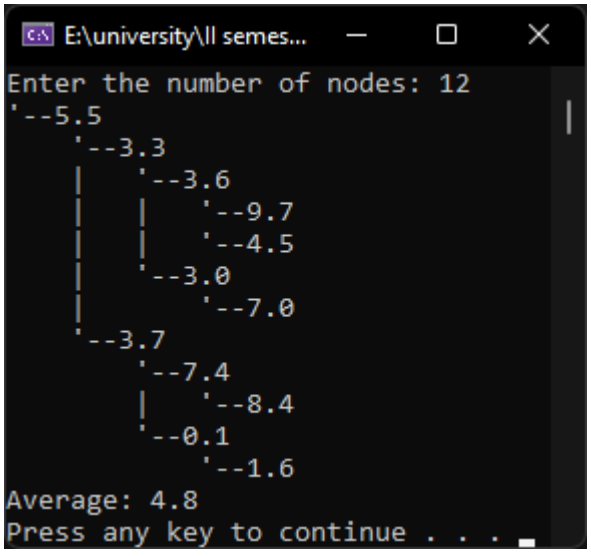
```cpp
#include "Tree.h"
#include <iomanip>
#include <queue>

 //конструктор з параметром
Node::Node(float value)
{
    data = value;
    left = nullptr;
    right = nullptr;
}

//вставлення нової вершини-нащадка
//pos == 1 -> left, pos == 2 -> right
Node* Node::insertNodePointer(Node* value, int pos)
{
    switch (pos)
    {
    case 1:
        left = value;
        return left;
    case 2:
        right = value;
        return right;
    }
    return nullptr;
}
```

```cpp
 //заповнення дерева заданими значеннями масиву
Node* Tree::createTree(float* arr, int from, int num)
{
    if (num == 0)
        return nullptr;
    Node* p = new Node(arr[from]);
    int numleft = num / 2;
    int numright = num - numleft - 1;
    p->insertNodePointer(createTree(arr, from + 1, numleft), 1);
    p->insertNodePointer(createTree(arr, from + 1 + numleft, numright), 2);
    return p;
}
```

```cpp
//знаходження суми значень вершин дерева
float Tree::getAverageBFS()
{
    std::queue <Node*> plan; //черга для пошуку в ширину
    Node* temp = root;       //проміжна змінна
    plan.push(temp);         //додання кореню дерева в чергу
    float sum = 0;           //сума значень
    float count = 0;         //кількість вершин
    Node* temp2;             //друга проміжна змінна
    while (plan.size()>0) //поки в черзі є елементи
    {
        //діставання елемента з черги
        temp = plan.front();
        plan.pop();

        //додавання лівого нащадка до черги
        temp2 = temp->getLeftPointer();
        if (temp2!=nullptr)
            plan.push(temp2);

        //додавання правого нащадка до черги
        temp2 = temp->getRightPointer();
        if (temp2 != nullptr)
            plan.push(temp2);

        //додавання значення temp до загальної суми
        if(temp!=nullptr)
            sum += temp->getData();

        count++;
    }
    return sum/count;
}
```

```cpp
//генерація масиву дійсних чисел заданого розміру
float* arrGen(int num)
{
    float* arr = new float[num];
    for (int i = 0; i < num; i++)
        arr[i] = float(rand() % 100) / 10;
    return arr;
}

//виведення дерева
void treeOutput(const std::string& prefix, Node* node, bool isLeft)
{
    if (node != nullptr)
    {
        std::cout << prefix;

        std::cout << "'--";

        std::cout << std::fixed << std::setprecision(1) << node->getData() << std::endl;

        treeOutput(prefix + (isLeft ? "|   " : "    "), node->getLeftPointer(), true);
        treeOutput(prefix + (isLeft ? "|   " : "    "), node->getRightPointer(), false);
    }
}

//введення кількості елементів дерева
int numInput()
{
    int num;
    std::cout << "Enter the number of nodes: ";
    std::cin >> num;
    return num;
}
```

*Результат:*



**Висновок:** При виконанні лабораторної роботи було вивчено особливості організації і обробки дерев та було програмно реалізовано цю структуру даних на мові програмування C++.