**Державний вищий навчальний заклад**

**Ужгородський національний університет**

**Факультет інформаційних технологій**

**Практична робота №6**

**Алгоритми на деревах**

Виконав студент 1 курсу

спеціальності “Інженерія

програмного забезпечення”

Гарабаджіу Крістіан

**Ужгород-2025**

**Мета:** набути навичок створення та обробки дерев.

Завдання до роботи:

1. Написати код програми відповідно до обраного варіанту. Кожен варіант містить завдання та спосіб обходу дерева для кожного із завдань.
2. Оформити звіт та завантажити звіт в системі електронного навчання ДВНЗ «УжНУ» в установлений термін.
3. Підготувати відповіді на контрольні питання.

**Хід роботи:**

Обхід дерева: прямий  
 1. Знайти суму всіх елементів дерева

2. Вивести на екран всі листи дерева

class brick{

constructor(vl, rt, rh, lh, lvl){

this.value = vl;

this.root = rt;

this.rightHand = rh;

this.leftHand = lh;

this.level = lvl;

}  
}  
class tree{

constructor(maxLvl){

this.start = new brick(this.generator(), null, null, null, 0);

this.maxLvl = maxLvl;

this.randomTree(this.start, maxLvl);

this.mainAr = [];

for (let i = 0; i < maxLvl; i++) {this.mainAr.push([]);}

this.toAr(this.start, 0);

this.task1 = 0;

this.task2 = [];

}

randomTree(br, maxLvl){

if(br.level != maxLvl - 1){

if(br.value != "X"){

switch(Math.floor(Math.random() \* 10)){

case 0:

case 1:

case 2:

case 3:

case 4:

if(Math.floor(Math.random() \* 10 / 5) == 0){

br.rightHand = new brick("X", br, null, null, br.level + 1);

this.randomTree(br.rightHand, maxLvl);

br.leftHand = new brick(this.generator(), br, null, null, br.level + 1);

this.randomTree(br.leftHand, maxLvl);

} else {

br.rightHand = new brick(this.generator(), br, null, null, br.level + 1);

this.randomTree(br.rightHand, maxLvl);

br.leftHand = new brick("X", br, null, null, br.level + 1);

this.randomTree(br.leftHand, maxLvl);

}

break;

case 5:

case 6:

case 7:

br.leftHand = new brick(this.generator(), br, null, null, br.level + 1);

this.randomTree(br.leftHand, maxLvl);

br.rightHand = new brick(this.generator(), br, null, null, br.level + 1);

this.randomTree(br.rightHand, maxLvl);

break;

case 8:

case 9:

br.leftHand = new brick("X", br, null, null, br.level + 1);

this.randomTree(br.leftHand, maxLvl);

br.rightHand = new brick("X", br, null, null, br.level + 1);

this.randomTree(br.rightHand, maxLvl);

break;

}

}else{

br.leftHand = new brick("X", br, null, null, br.level + 1);

this.randomTree(br.leftHand, maxLvl);

br.rightHand = new brick("X", br, null, null, br.level + 1);

this.randomTree(br.rightHand, maxLvl);

}

}

}

generator(){

return Math.floor(Math.random() \* (100 - 0 + 1));

}

toAr(br, prevLevel){

if (br != null) {

this.mainAr[br.level].push(br.value);

if (br.leftHand != null) {

this.toAr(br.leftHand, br.level);

} else if (br.level + 1 < this.maxLvl) {

this.mainAr[br.level + 1].push("X");

}

if (br.rightHand != null) {

this.toAr(br.rightHand, br.level);

} else if (br.level + 1 < this.maxLvl) {

this.mainAr[br.level + 1].push("X");

}

}  
}

seeTree(){

var rowLength = Math.pow(2, this.maxLvl) \* 3;

for (let i = 0; i < this.maxLvl; i++) {

var space = (" ").repeat(rowLength / (Math.pow(2, i) + 1) - 1);

var space2 = (" ").repeat(rowLength / (Math.pow(2, i) + 1) - 2);

var finRow = [];

finRow.push(space)

for(let a = 0; a < Math.pow(2, i); a++){

var ob = this.mainAr[i][a] == "X" ? " " : (this.mainAr[i][a] / 10 < 1 ? " " + this.mainAr[i][a] : this.mainAr[i][a]);

finRow.push(ob);

finRow.push(space2);

}

console.log(finRow.join(''));

}

}

taskRes(br) {

this.task1 += br.value;

if (br.leftHand && br.leftHand.value !== "X") {

this.taskRes(br.leftHand);

}

if (br.rightHand && br.rightHand.value !== "X") {

this.taskRes(br.rightHand);

}

if ((!br.leftHand || br.leftHand.value === "X") && (!br.rightHand || br.rightHand.value === "X")) {

this.task2.push(br.value);

}

}

taskResConsol(){

this.taskRes(this.start);

console.log(`Сума елементів дерева: ${this.task1}.`);

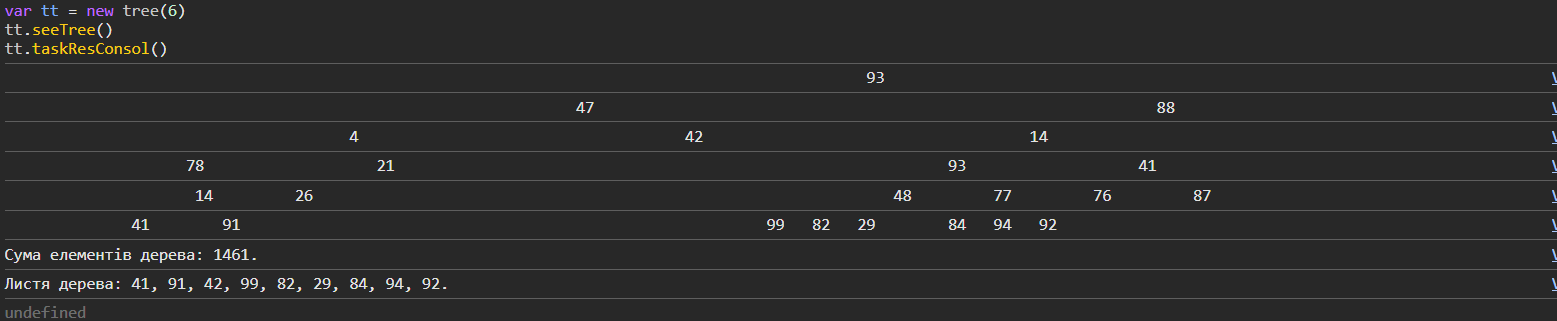
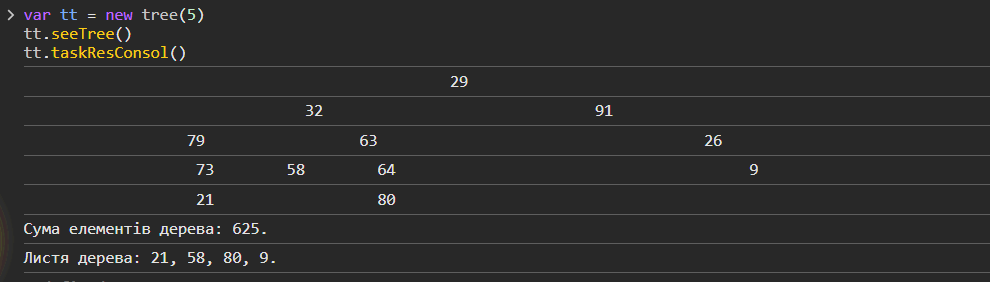
console.log(`Листя дерева: ${this.task2.join(", ")}.`);

}  
}

var tt = new tree(5)

tt.seeTree()

tt.taskResConsol()



**Висновок:**

У результаті виконання завдання було сформовано бінарне дерево за допомогою класів brick і tree, де кожен вузол містить значення, рівень і посилання на батьківський та дочірні елементи. Генерація дерева відбувається рекурсивно з урахуванням заданої максимальної глибини (maxLvl), а значення вузлів випадкові, крім спеціального маркера "X", що позначає відсутність підгілки. Застосування прямого (префіксного) обходу (taskRes) дало змогу реалізувати підрахунок суми всіх числових значень у дереві, виключаючи "X", а також знайти всі листові вузли — тобто ті, що не мають жодного нащадка. Дерево наочно виводиться у вигляді псевдографіки через метод seeTree, що спрощує візуальне сприйняття його структури. Програма демонструє ефективну рекурсивну обробку складних деревоподібних структур та дозволяє вивчити важливі алгоритми обходу. Крім того, правильна організація умов у randomTree дозволяє динамічно формувати як повні, так і неповні дерева з різною конфігурацією. Робота сприяє розумінню принципів побудови, візуалізації та обробки дерев у програмуванні.