**Державний вищий навчальний заклад**

**Ужгородський національний університет**

**Факультет інформаційних технологій**

**Практична робота № 6**

**Тема:** Алгоритми на деревах.

Виконав студент І курсу

Спеціальності «Інженерія

програмного забезпечення»

Боднарчук Роман Романович

**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 його в системі електронного навчання ДВНЗ «УжНУ» в установлений термін.
3. Підготувати відповіді на контрольні питання.

# Хід роботи

**Варіант 1**

**Обхід дерева:** Прямий.

**Завдання:**

* 1. Знайти суму всіх елементів дерева
  2. Вивести на екран всі листи дерева

1. function TreeNode(val) {
2. this.val = val;
3. this.left = this.right = null;
4. }
6. function sumTree(root) {
7. if (root === null) {
8. return 0;
9. }
10. return root.val + sumTree(root.left) + sumTree(root.right);
11. }
13. function findLeaves(root) {
14. let leaves = [];
15. function traverse(node) {
16. if (node !== null) {
17. if (node.left === null && node.right === null) {
18. leaves.push(node.val);
19. }
20. traverse(node.left);
21. traverse(node.right);
22. }
23. }
24. traverse(root);
25. return leaves;
26. }
28. let root = new TreeNode(1);
29. root.left = new TreeNode(2);
30. root.right = new TreeNode(3);
31. root.left.left = new TreeNode(4);
32. root.left.right = new TreeNode(5);
34. console.log("Сума всіх елементів дерева:", sumTree(root));
36. console.log("Листи дерева:", findLeaves(root));



**Питання для самоконтролю**

1. З чим пов’язана популярність використання дерев у програмуванні?

**Відповідь:** Дерева широко використовуються в програмуванні завдяки своїм структурним властивостям, які дозволяють ефективно вирішувати різноманітні задачі. Дерева дозволяють швидко вставляти, видаляти та знаходити елементи, а також вони використовуються для представлення ієрархічних даних, таких як файлові системи, структури документів та інші.

1. Чи можна список віднести до дерев? Відповідь обґрунтуйте.

**Відповідь:** Так, список можна вважати деревом з одним кореневим вузлом та вузлами-нащадками, які посилаються один на одного за допомогою посилань. Кожен елемент списку може мати посилання на наступний елемент у списку, утворюючи ланцюжок. Така структура схожа на просте бінарне дерево, де кожен вузол може мати два нащадки.

1. Які дані містять адресні поля елемента бінарного дерева?

**Відповідь:** Елемент бінарного дерева містить адресні поля, які зазвичай називаються "вказівниками" (pointers) або "посиланнями" (references). Вони вказують на лівого та правого нащадка елемента. Також може бути вказівник на батьківський вузол, якщо потрібно.

1. Чи може бінарне дерево бути строгим і неповним? Відповідь обґрунтуйте.

**Відповідь:** Так, бінарне дерево може бути строгим і неповним одночасно. Строге дерево - це дерево, у якому кожен вузол має або нуль, або два нащадки. Неповне дерево - це дерево, у якому деякі вузли можуть мати лише одного або жодного нащадка. Такі дерева є допустимими і можуть виникати в результаті деяких операцій, таких як видалення вузлів або конструкція дерева з неповних даних.

1. Чи може бінарне дерево бути нестрогим і повним? Відповідь обґрунтуйте.

**Відповідь:** Так, бінарне дерево може бути нестрогим і повним. Нестроге дерево - це дерево, у якому кожен вузол може мати від 0 до 2 нащадків. Повне дерево - це дерево, у якому всі рівні (крім можливо останнього) заповнені повністю, тобто кожен вузол має два нащадки, а останній рівень заповнений зліва направо без пропусків. Таке дерево може існувати, наприклад, якщо до існуючого строгого дерева додається новий елемент, що має тільки одного нащадка.

1. Яким може бути майже збалансоване бінарне дерево: повним, неповним, строгим, нестрогим? Відповідь обґрунтуйте.

**Відповідь:** Майже збалансоване бінарне дерево може бути неповним або повним, але нестрогим. Збалансоване дерево - це дерево, у якому різниця у висоті піддеревин на будь-якому рівні не перевищує одиницю. Якщо дерево майже збалансоване, це означає, що висота лівого та правого піддерев різна не більше, ніж на одиницю, але може бути рівною. Таке дерево може бути неповним, якщо деякі гілки закінчуються на різних рівнях, або повним, якщо всі рівні, крім можливо останнього, заповнені. Нестроге дерево не може бути майже збалансованим, оскільки для збалансованості на кожному рівні потрібно, щоб кожен вузол мав або два нащадки, або жодного.

1. Куди може бути доданий елемент у бінарне дерево у залежності від його виду (повне, неповне, строгим, нестрогим)? Вид дерева при цьому повинен зберегтися.

**Відповідь:** У залежності від виду бінарного дерева:

* У повному бінарному дереві, яке є також строгим, новий елемент додається на наступному рівні після останнього вузла, який має один або жодного нащадка.
* У неповному бінарному дереві, яке може бути як строгим, так і нестрогим, новий елемент додається так само як у повному дереві, але може займати будь-яке вільне місце на останньому рівні.

У обох випадках, важливо зберегти властивість бінарного дерева, де кожен вузол може мати не більше двох нащадків.

1. Куди може бути доданий елемент у збалансоване бінарне дерево? Вид дерева при цьому повинен зберегтися.

**Відповідь:** У збалансоване бінарне дерево можна додати елемент у такий спосіб, щоб після додавання дерево залишалося збалансованим. Для цього можна використовувати різні методи балансування, наприклад, методи AVL-дерева, червоно-чорного дерева тощо. В цих методах додавання елемента відбувається так, щоб підтримувати баланс дерева, зберігаючи його структуру та властивості.

1. Чим відрізняються, з точки зору реалізації алгоритму прямий, симетричний і зворотний обходи бінарного дерева?

**Відповідь:** Прямий (pre-order), симетричний (in-order) і зворотний (post-order) обходи бінарного дерева відрізняються порядком обробки вузлів:

* Прямий обхід: спочатку відвідується корінь, потім лівий нащадок, і нарешті правий нащадок. Цей обхід використовується для виведення вузлів дерева у вигляді префіксного виразу.
* Симетричний обхід: спочатку відвідується лівий нащадок, потім корінь, і нарешті правий нащадок. Цей обхід використовується для виведення вузлів дерева у вигляді інфіксного виразу.
* Зворотний обхід: спочатку відвідується лівий нащадок, потім правий нащадок, і нарешті корінь. Цей обхід використовується для виведення вузлів дерева у вигляді постфіксного виразу.

Реалізація кожного з цих алгоритмів відрізняється в залежності від того, як вибрано структуру даних для представлення бінарного дерева (наприклад, рекурсивно або ітеративно) та в якому порядку вузли обробляються (наприклад, за допомогою стеку або рекурсії).

**Висновки:** У ході виконання Практичної роботи №6, я написав алгоритм на деревах. Алгоритми на деревах використовуються для обробки і структурування інформації у вигляді деревоподібної структури. Основні операції включають пошук, вставку, видалення елементів, а також обходи дерева для виводу або обробки елементів у певному порядку. Кожен з трьох основних обходів (прямий, симетричний, зворотний) має свої особливості та застосування. Реалізація алгоритмів на деревах може відрізнятися в залежності від типу дерева та мети використання. У своєму коді я зробив суму всіх елементів та вивів на екран всі листки дерева. Тим самим набув практичних навичок із виконанням алгоритмів на деревах та зробив висновок.