Лабораторна робота № 5
Дерева

Мета роботи: отримати навики застосування двійкових дерев, реалізувати основні операції над деревами: обхід дерев, включення, виключення та пошук вузлів.

5.1 Теоретичні відомості

Дерево - це нелінійна структура даних, що використовується для представлення ієрархічних зв'язків, що мають відношення «один до багатьох».

Дерево з базовим типом T визначається рекурсивно або як порожня структура (порожнє дерево), або як вузол типу T з кінцевою кількістю деревовидних структур цього ж типу, які називаються піддеревами.

Дерева використовуються при побудові організаційних діаграм, аналізі електричних ланцюгів, для представлення синтаксичних структур у компіляторах програм, для представлення структур математичних формул, організації інформації в СУБД тощо.

Самий верхній вузол дерева називається коренем. Верхній вузол для нижнього вузла називається предком, а нижній вузол для верхнього - нащадком. Вершини (вузли), що не мають нащадків, називаються листками дерева. Вершини, що мають нащадків, називаються внутрішніми. Дві вершини дерева з'єднуються гілкою. Кількість гілок від кореня до вершини є довжиною шляху до цієї вершини.

Довжина шляху від кореня до будь-якої вершини називається глибиною цієї вершини. Максимальна глибина вершин дерева називається висотою дерева.

Кількість безпосередніх нащадків у вершини (вузла) дерева називається степенем вершини (вузла). Максимальний степінь всіх вершин є степенем дерева.

Кожному вузлу дерева можна співставити ім'я вузла і значення вузла, тобто дані, які зберігаються в цьому вузлі. Причому, якщо значенням є різнорідні дані (записи або об'єднання), то значенням вузла можна вважати значення одного з полів цих даних, яке називається ключем.

Наприклад, у дерева на рис. 1 номер вузла може бути як його ім'ям, так і його значенням.

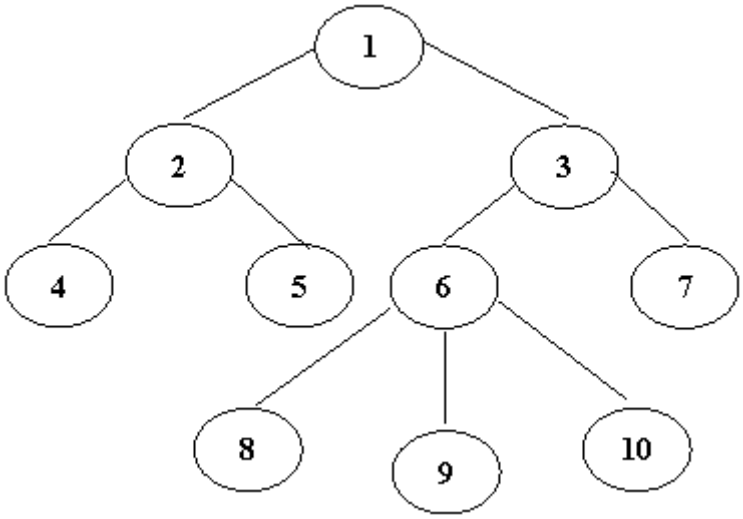

Рис. 1

У пам'яті дерева можна представити у вигляді зв'язків з предками (батьками); зв'язного списку нащадків (дітей) або структури даних.

Подання зазначеного дерева (див. рис. 1) у вигляді зв'язків з предками:

№ вершина	1	2	3	4	5	6	7	8	9	10
Батько	0	1	1	2	2	3	3	6	6	6

Приклад подання цього ж дерева у вигляді зв'язного списку нащадків наведено на рис 2:

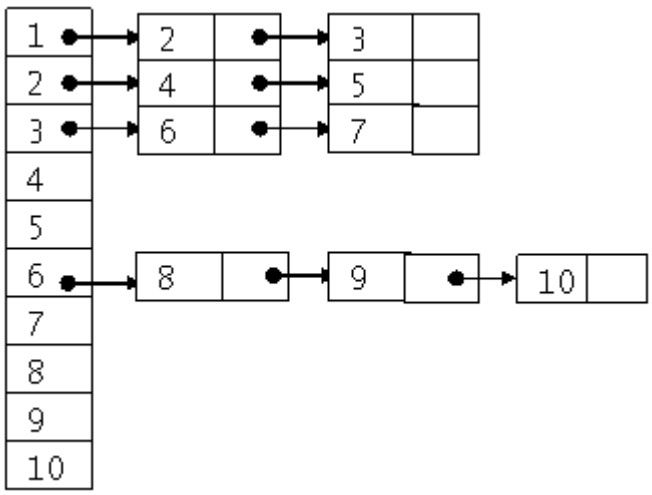

Рис. 2.

Якщо у кожної вершини дерева є не більше двох нащадків (ліві і праві піддерева), то таке дерево називається двійковим або бінарним.

Двійкові дерева широко використовуються у програмуванні.

Основні операції з деревами: обхід дерева, пошук по дереву, включення в дерево, виключення з дерева.

Обхід (відвідування) вершин дерева можна здійснити наступним чином (рис. 3):

- Зліва направо: A, R, B (інфіксний обхід, симетричний обхід) .
- Зверху вниз: R, A, B (префіксній обхід, обхід в прямому порядку).
- Знизу вверх: A, B, R (постфіксний обхід, обхід в зворотному порядку).

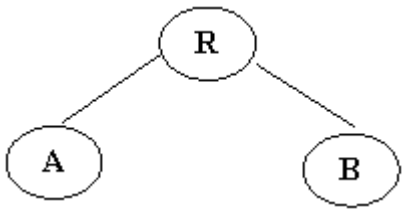

Рис. 3

Для реалізації алгоритмів пошуку використовуються дерева двійкового пошуку. Дерево двійкового пошуку - це таке дерево, в якому всі ліві нащадки молодші за предка, а всі праві - старші. Ця властивість називається характеристичною властивістю дерева двійкового пошуку і виконується для будь-якого вузла, включаючи корінь. З урахуванням цієї властивості пошук вузла в двійковому дереві пошуку можна здійснити, рухаючись від кореня в ліве або праве піддерево в залежності від значення ключа піддерева.

Якщо при побудові дерева по черзі розташовувати вузли зліва та справа, то вийде дерево, у якого кількість вершин у лівому та правому піддеревах відрізняється не більше ніж на одиницю. Таке дерево називається ідеально збалансованим.

N елементів можна організувати в бінарне дерево з висотою не більше $\log_2(N)$, тому для пошуку серед N елементів може знадобитись не більше $\log_2(N)$ порівнянь, якщо дерево ідеально збалансоване. Звідси випливає, що дерево - це одна з найкращих структур для організації пошуку.

Операція включення елемента у дерево розбивається на три етапи: включення вузла в порожнє дерево, пошук кореня для додавання нового вузла, включення вузла в ліве або праве піддерево.

Для видалення вузла з зазначеним ключем спочатку відбувається його пошук. У випадку, якщо вузол знайдений, то він видаляється. При цьому розглядаються наступні три випадки. Якщо видаляється вузол, у якого немає нащадків, то просто вказівнику на нього присвоюється значення NULL. Якщо видаляється вузол, який має одного нащадка, в цьому випадку переадресується вказівник на цього нащадка. Якщо видаляється вузол, який має двох нащадків, то на його місце ставиться самий лівий нащадок з правого піддерева.

5.2. Порядок виконання роботи

Побудувати дерево відповідно до варіанту. Вивести його на екран у вигляді дерева. Реалізувати основні операції роботи з деревом: обхід дерева, включення, виключення та пошук вузлів. Оформити їх у вигляді функцій.

При розробці інтерфейсу програми слід передбачити:

- вказати тип, формат і діапазон даних, що вводяться;
- вказати дії, що здійснює програма;
- наявність пояснень при виведенні результату;
- відображення дерева візуалізувати.

Під час тестування програми необхідно:

- перевірити правильність введення і виведення даних (тобто їх відповідність необхідному типу і формату). Забезпечити адекватну реакцію програми на невірне введення даних;
- забезпечити виведення повідомлень за відсутності вхідних даних («порожнє введення»);
- перевірити правильність виконання операцій;
- забезпечити можливість додавання вузла в порожнє дерево;
- передбачити відображення повідомлення при спробі видалити вузол з порожнього дерева;
- перевірити різні випадки включення і виключення вузла в існуючому дереві;
- перевірити пошук існуючого вузла та пошук неіснуючого вузла в дереві.

5.3. Варіанти завдань

На оцінку **«добре»:**

1. Побудувати двійкове дерево пошуку з цілих чисел, що вводяться. Вивести його на екран у вигляді дерева. Знайти вершину, яка містить задане число. Визначити максимальний елемент в цьому дереві.

На оцінку **«відмінно»:**

1. Побудувати двійкове дерево пошуку з букв рядка, що вводиться. Вивести його на екран у вигляді дерева. Знайти букви, що зустрічаються більше одного разу. Видалити з дерева ці літери. Вивести елементи дерева, що залишилися, при його постфіксному обході.

2. Побудувати двійкове дерево пошуку, в вершинах якого знаходяться слова з текстового файлу. Вивести його на екран у вигляді дерева. Визначити кількість вершин дерева, що містять слова, які починаються на зазначену букву. Видалити з дерева ці вершини.

3. Побудувати словник зі слів текстового файлу у вигляді дерева двійкового пошуку. Вивести його на екран у вигляді дерева. Здійснити пошук вказаного слова у дереві і у файлі. Якщо слова немає, додати його при

необхідності в дерево і у відповідний файл. Видалити вказане слово з дерева та файлу. Порівняти час пошуку в дереві та у файлі.