Міністерство освіти і науки України
Львівський національний університет імені Івана Франка
Факультет прикладної математики та інформатики

Кафедра дискретного аналізу та інтелектуальних систем

Лабораторна робота №11

AVL ДЕРЕВА

з курсу “Алгоритми та структури даних”

Виконав:
студент групи ПМІ-13
Лук’янчук Денис
Євгенійович

Львів – 2024

Дерево AVL - це вид бінарного дерева пошуку, який використовується для зберігання та швидкого пошуку даних. Основна відмінність дерева AVL від звичайного бінарного дерева полягає в тому, що воно забезпечує балансування висоти піддерев для забезпечення оптимального часу доступу до даних. У дереві AVL для кожного вузла виконується умова балансу, яка гарантує, що різниця висоти його лівого та правого піддерев не перевищує 1.

Основні властивості дерева AVL:

1. **Балансування**: Для кожного вузла дерева виконується умова балансу, що означає, що висоти його лівого та правого піддерев не можуть відрізнятися більш ніж на одиницю.

2. **Самобалансуючість**: Після вставки або видалення вузла з дерева, його структура автоматично коригується таким чином, щоб забезпечити виконання умови балансу.

3. **Швидкий доступ**: Завдяки умові балансу дерево AVL гарантує оптимальний час доступу до даних, який становить O(log n), де n - кількість вузлів у дереві.

4. **Оптимальне використання пам'яті**: Дерева AVL зберігаються у вигляді збалансованих структур, що дозволяє ефективно використовувати пам'ять.

Алгоритм виконання:

1. Структура `Node` визначає вузол дерева з полями `key` (ключ), `left` (ліве піддерево), `right` (праве піддерево) та `height` (висота вузла).

2. Функція `newNode` створює новий вузол з заданим ключем.

3. Функція `height` повертає висоту вузла.

4. Функції `rightRotate` та `leftRotate` виконують праве та ліве обертання відповідно для балансування дерева.

5. Функція `printAVL` виводить дерево у вигляді, зручному для розуміння, використовуючи обхід у порядку `правий-корінь-лівий`.

6. Функція `minValueNode` знаходить вузол з мінімальним ключем у піддереві.

7. Функція `deleteNode` видаляє вузол з заданим ключем з дерева.

8. Функція `insert` вставляє новий вузол з заданим ключем у дерево.

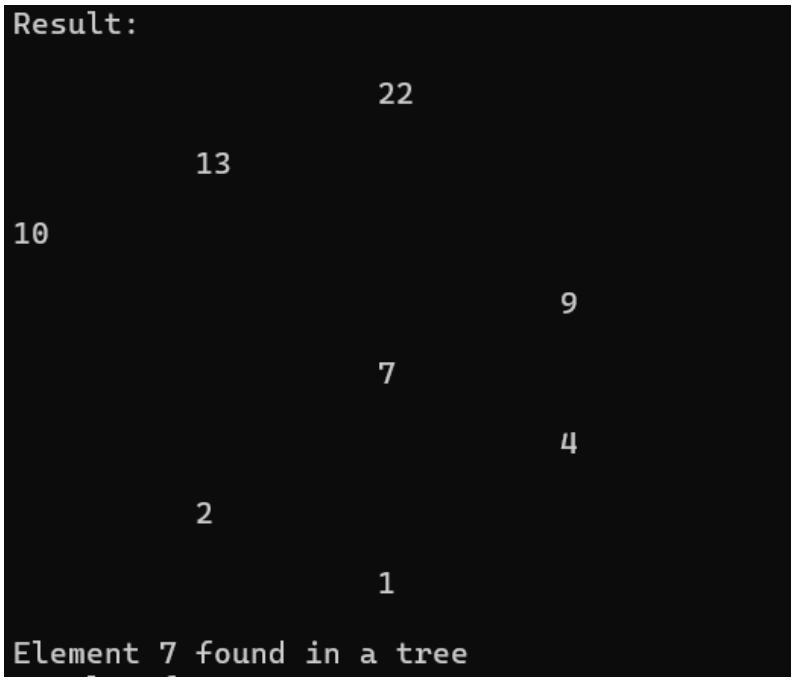
9. Функція `search` виконує пошук заданого ключа у дереві.

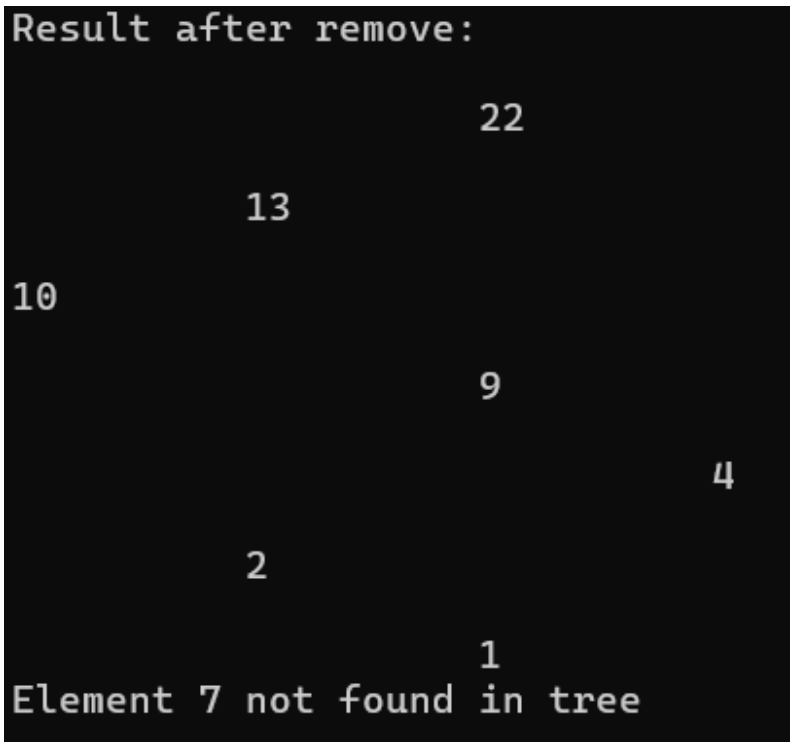
10. У функції `main` виконується наступне:

- Створюється пусте дерево `root`.
- Виконується послідовна вставка деяких ключів у дерево.
- Виводиться дерево за допомогою функції `printAVL`.
- Виконується пошук елемента у дереві.
- Видаляється елемент з ключем 7.
- Повторно виводиться дерево.
- Знову виконується пошук того ж елемента.

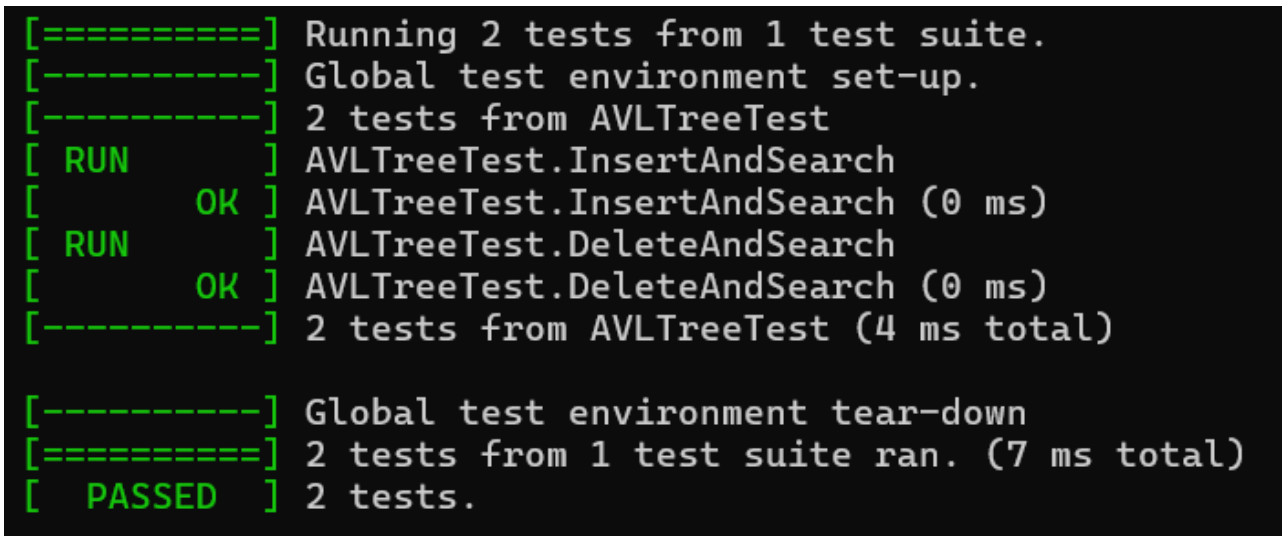
Приклад

Дано: elements = {10, 2, 13, 22, 1, 9, 7, 4}

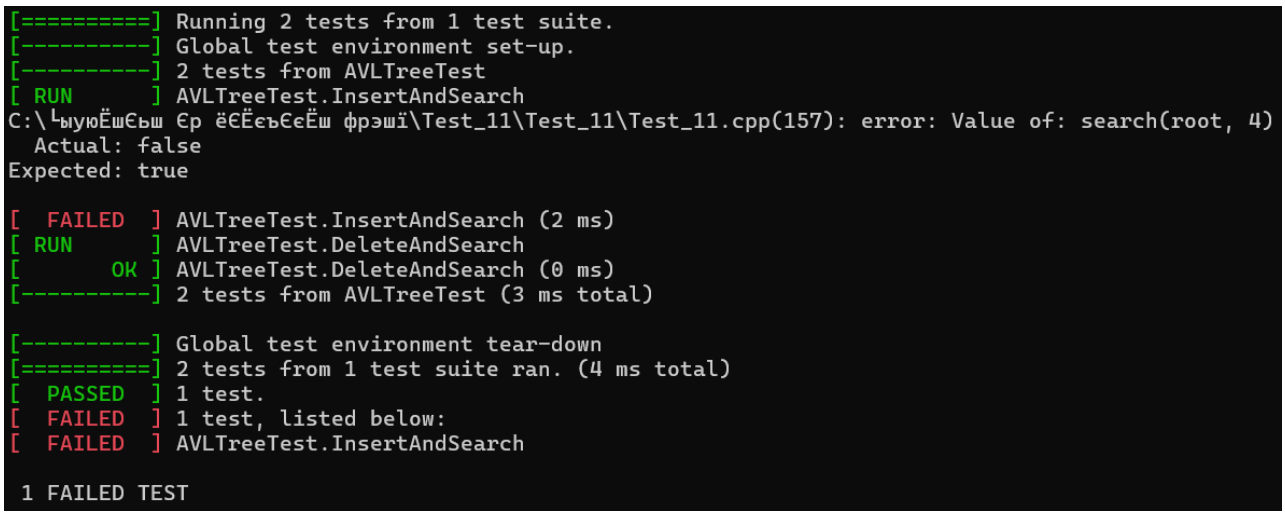




Приклад Unit-тесту(*без помилок*)



Приклад Unit-тесту(*з помилкою*)



Висновок: Наведений код реалізує операції вставки, пошуку та видалення вузлів у дереві пошуку AVL. Дерево пошуку AVL є важливою структурою даних, оскільки воно гарантує балансування, що забезпечує швидкий доступ до даних у середньому та найгірший час доступу за O(log n), де n - кількість елементів у дереві. Кожна операція вставки, пошуку та видалення вузла має складність O(log n), де n - кількість елементів у дереві, завдяки тому, що AVL-дерева підтримують балансування висоти піддерев. Це гарантує ефективну роботу з даними навіть у великих обсягах. Таким чином, використання дерева пошуку AVL є вигідним для задач, де потрібно забезпечити швидкий пошук, вставку та видалення елементів з мінімальним часом виконання.