|  |  |  |  |
| --- | --- | --- | --- |
| МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ  КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ імені Тараса Шевченка  ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ  **Кафедра програмних систем і технологій**  Дисципліна  **«Структури данних, аналіз і**  **алгоритми ком’ютерної**  **обробки інформаціїї»**  **Лабораторна робота № 2**  **на тему:**  **Бінарні Дерева** | | | |
| **Виконав:** | Аврущенко Дмитро Мансоровича | **Перевірила**: | Бичков Олексій Сергійович |
| Група | ІПЗ-12(1) | Дата перевірки |  |
| Форма навчання | денна | Оцінка |  |
| Спеціальність | 121 |
| 2022 | | | |

**Умова лабораторної:**

Використовуючи структуру даних: бінарне дерево пошуку, червоно-чорне дерево та АВЛ дерево реалізувати такі алгоритми:

1. Вставка вузла в дерево
2. Видалення вузла із дерева
3. Перефарбування двох однакових вузлів братів у протилежний колір
4. Лівий, правий повороти
5. Splay алгоритм?

**Аналіз**

Бінарні дерева – структура данних, в якому кожен елемент має значення та посилання на батьківського лівого та правого елемента. Також воно має свої властивості, а саме лівий елемент повинен бути менше поточного та правий елемент – більше. Також бінарні дерева поділяються на Червоно-чорні та АВЛ дерева, які є само балансуючими. Червоно-чорні мають такі властивості (кожна вершина або червона, або чорна, корінь дерева — чорний, кожний лист вважаємо чорним, якщо вершина червона, обидві її дочірні вершини чорні (інакше, батько червоної вершини — чорний), усі прості шляхи від будь-якої вершини до листів мають однакову кількість чорних вершин), які треба реалізувати. АВЛ дерева реалізовує такі ж властивості, що і бінарні дерева пошуку, але має ще атрибут висота, який у кожного піддерева має на одному рівні не відрізнятися на 1 одиницю. Splay алгоритм реалізується на бінарному дереві пошуку, але кожен новий елемент стає на місце першого, також елемент під час його пошуку передається у вершину, тому всі елементи, що частіше використовуються будуть ближче до кореня дерева.

**Структура основних вхідних та вихідних даних**

* Вхідні дані:

Вхідні дані – заготовленні числа для того, щоб зрозуміти як працюють дерева, виконуючі свої властивості

* Вихідні дані:

Отримання дерева у якому є всі вхідні данні, які не порушують властивості

**Алгоритм розв’язання задачі:**

Для того, щоб виконати це завдання нам необхідно прочитати теорію.

Для бінарного дерева пошуку був створений лише один клас без допоміжних класів Node, які були використані в інших. Всі дерева складаються з об’єкту(інформації) та посилання на батьківський, лівий та правий вузол. Всі дерева, крім звичайних бінарних, мають посилання на корінний елемент, для простого оперування поворотами.

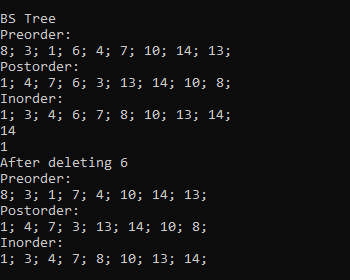
Бінарне дерево пошуку реалізуємо так: якщо елемент менше поточного, то йдемо в ліве піддерево, якщо більше – праве піддерево. Майже нічого складного

Червоно-чорне дерево реалізуємо так: спочатку як бінарне дерево пошуку, потім зберігаємо колір поточного елемента та виконуємо балансування, перевіряючи властивості ( кожна вершина або червона, або чорна, корінь дерева — чорний, кожний лист вважаємо чорним, якщо вершина червона, обидві її дочірні вершини чорні (інакше, батько червоної вершини — чорний), усі прості шляхи від будь-якої вершини до листів мають однакову кількість чорних вершин).

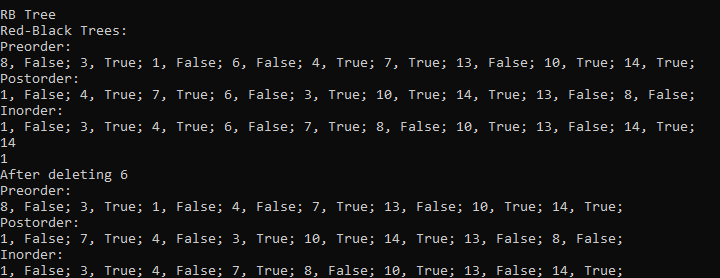
АВЛ дерево реалізуємо так: спочатку як бінарне дерево пошуку, потім зберігаємо висоту поточного елемента від листа. Якщо є різниця між висотами лівого та правого піддерева більше ніж одиниця, то починаємо балансувати за допомогою відомих формул.

Splay алгоритм я вирішив реалізувати в бінарному дереві так як всі інші самобалансуючі, тобто якщо я насильно поставлю елемент у корінний елемент він посунеться для зберігання властивостей (я про Червоно-чорне та АВЛ дерева).

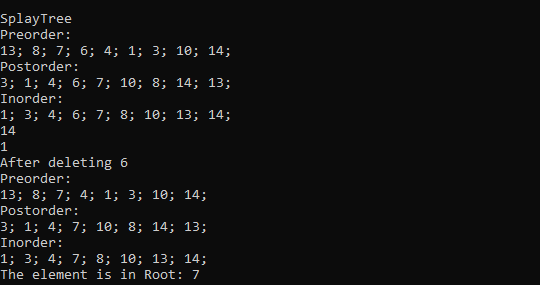
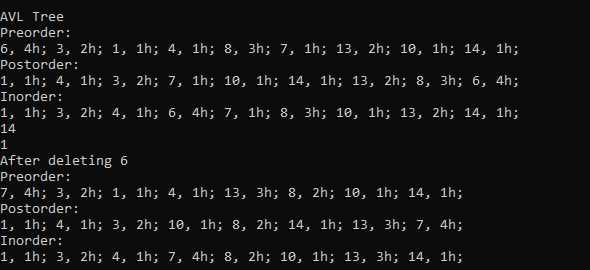
**Результати тестування**



Колір я реалізував як булеве значення True – Червоний, False – Чорне

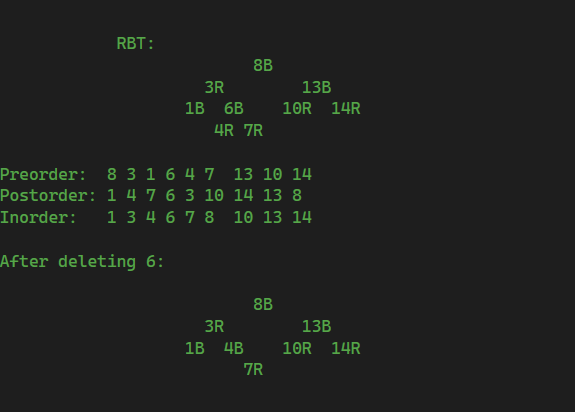
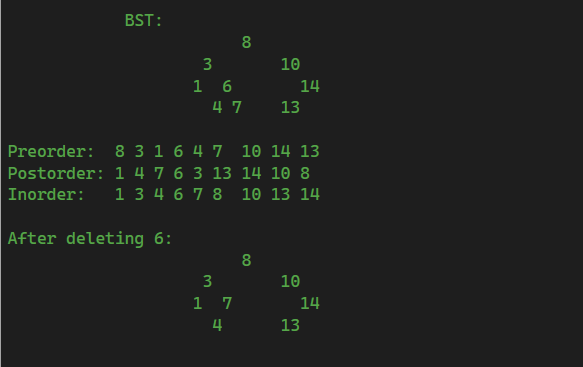


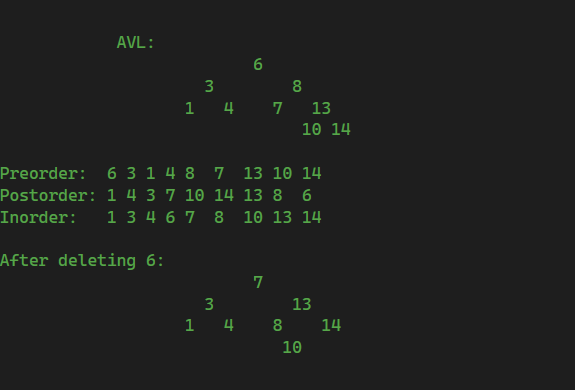
Висота має підпис h



**Аналіз отриманих результатів**

Перед всіма тестами програми та виводами я на листочку підготував очікуванні результати та можу дійти висновку, що програма працює як я задавав їй з початку.



Изображение выглядит как текст, электроника

Автоматически созданное описание

**Аналіз помилок**

Перші помилки з’являлися під час реалізації, бо у мене не було досить чіткого бачення як треба реалізовувати всі алгоритми, хоча на лекціях мені було все зрозуміло, доки я не взяв папір та не почав малювати дерева та алгоритми на них. Потім вже з’явилося розуміння, що треба робити та як.

Багато помилок в мене з’являлося із-за моєї недальновидності, а саме постійно звертався до пустих елементів, через батьків або синів, які теж були пустими, тому у мене була завжди помилка NullReferenceExeption, ці помилки також були прибрані після того як я взяв папір та почав малювати дерева.

Також багато разів я випадково видаляв багато елементів з дерева, коли робив ліві та праві повороти.

**Висновок**

Під час цієї лабораторної роботи я навчився користуватися бінарними деревами пошуку. Для мене найскладнішими були червоно-чорні, бо я не розумів повністю як їх потрібно реалізовувати. Найпростішими це звичайні бінарні дерева пошуку та Splay алгоритм(він був простим, бо до того часу я вже повністю реалізував все, що було необхідно)