Київський національний університет імені Тараса Шевченка

Факультет комп’ютерних наук та кібернетики

Кафедра інтелектуальних інформацій систем

Алгоритми та складніcть

Лабораторна робота №1

Завдання №2 – B-дерево

Тип даних – набір комлпексних чисел

Виконав студент 2-го курсу

Групи ПІ-23

Євчик Олексій

2023

***Зміст***

**Теоретичні відомості**

**Властивості**

**Принцип роботи**

**Алгоритм дерева**

**Складність**

**Мова програмування**

**Модулі програми**

**Інтерфейс користувача**

**Джерела**

**Теоретичні відомості**

B-дерева - узагальнення бінарних дерев пошуку.

Особливості:

• Робота з великими об’ємами даних, які не

поміщаються в оперативну пам’ять і зберігаються на

диску (наприклад, СУБД, файлові системи).

• Висока степінь розгалуження – вузли можуть мати

до тисяч потомків.

• Якщо внутрішній вузол містить n[x] ключів, то він має

(n[x]+1) синів.

• Ключі у вузлі x є роздільниками діапазону ключів на

(n[x]+1) піддіапазонів.

• При пошуку переходимо до сина з потрібним

діапазоном.

**Властивості**

Властивість 1: Глибина всіх листків однакова.

Властивість 2: Усі вузли, крім кореня, повинні мати як мінімум (m/2) – 1 ключів і максимум m-1 ключів.

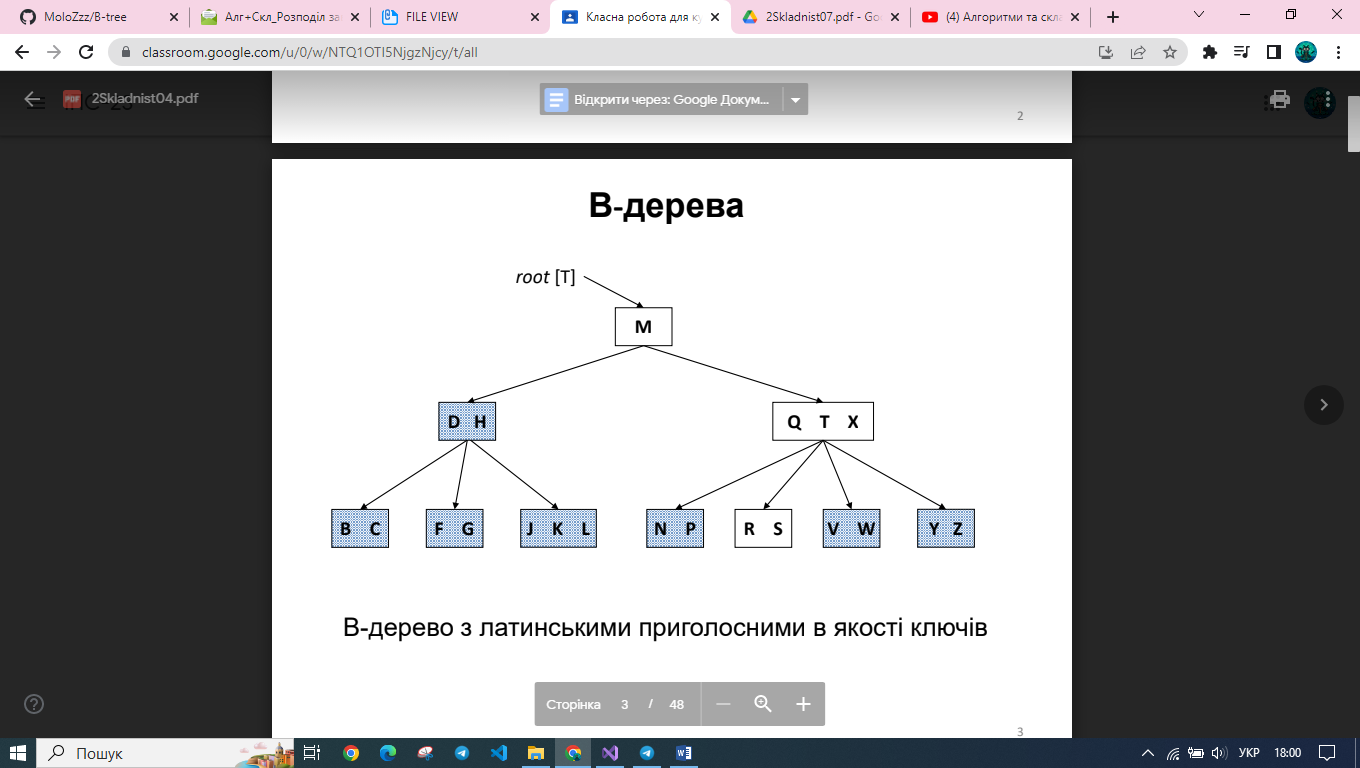
Властивість 3: Усі вузли без листя, крім кореня (тобто. всі внутрішні вузли), повинні мати мінімум m/2 нащадків.

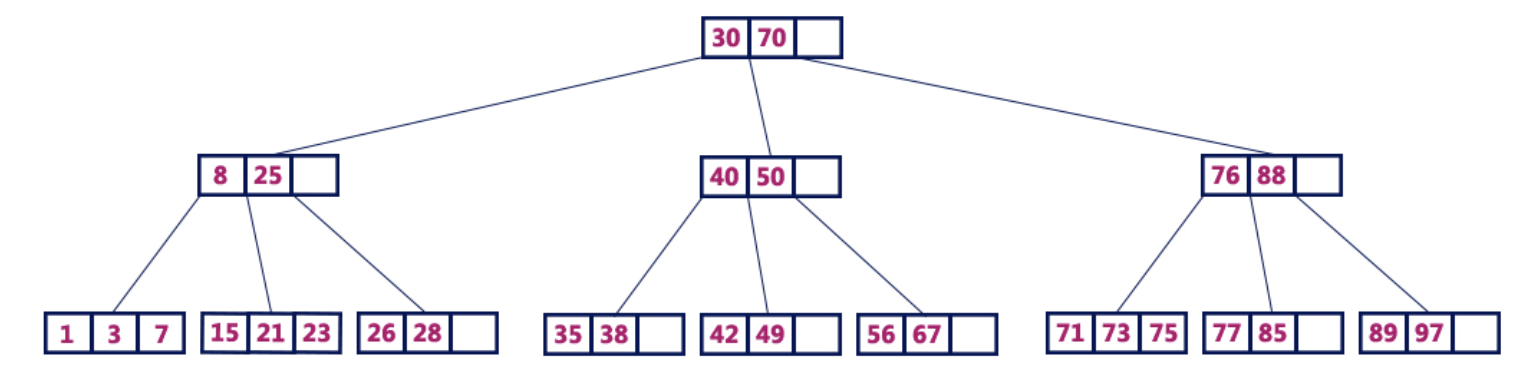
Властивість 4: Якщо корінь – це вузол, що не містить листя, він повинен мати мінімум 2 нащадки.

Властивість 5: Вузол без листя з n-1 ключами повинен мати n нащадків.

Властивість 6: Усі ключі у вузлі повинні розташовуватись у порядку зростання їх значень.

Зображення дерева:

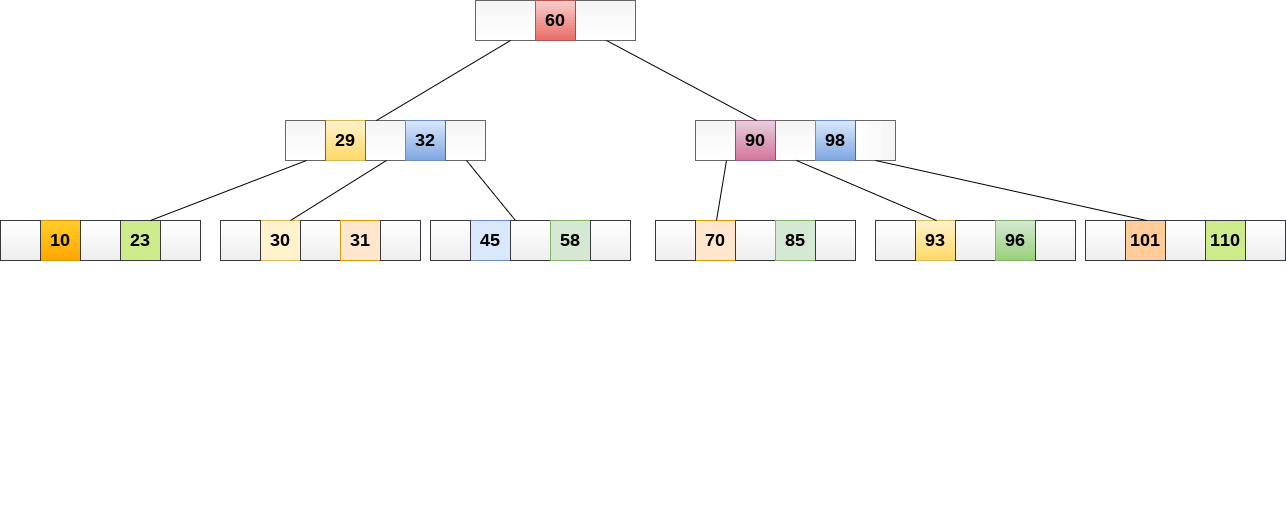




**Принцип роботи**

В-дерево схоже на повністю збалансоване бінарне дерево пошуку, але є багато відмінностей. В В-дереві кожен вузол може містити кілька значень(ключів) й довільну кількість синів, дерево завжди збалансоване(кожен рівень заповнений або пустий),це досягається за рахунок додавання значень у вузли.

Приклад: перший рівень містить 1 вузол(корінь, проте В-дерево росте знизу вверх), цей вузол має певну кількість значень(назвемо цю кількість M) і ця кількість може дорівнювати від 0 до М. Тобто на 1 рівні можлива кількість значень від 0 до М.

Тоді наступний рівень буде містити або 0, або M + 1 вузлів(а значення від 0 до (М+1)\*М тому що кількість вузлів множимо на можливу кількість значень у вузлі). Чому саме М+1 дітей? Тому що кожне значення у вузлі супроводжується 2 посиланнями(2 значення можуть мати одне посилання на сина), це посилання на вузол з меншими за дане значення число і з більшими за значення числами. Якщо вузол містить 1 значення то посилань на синів 2, якщо 3 то вже 3(так як посилання першого значення вказує на числа менші за нього й на більші, проте менші за наступне значення у вузлі…), 4 значення – 5 – посилань і так далі, схематично: 

Як бачимо, ліве(умовно) посилання в кореня посилається на вузол з меншими за нього значеннями, а правий – більшими, розглянемо його лівого сина, вузол містить 3 значення, й його посилання так само посилаються на вузли з більшими і меншими значеннями. Максимальну кількість значень у вузлі обираємо самостійно.

**Алгоритм дерева**

Над B-деревом можна проводити такі операції: пошук, вставка, вилучення.

***Пошук по B-дереву:***

Пошук по B-дереву аналогічний пошуку по двійковому дереву пошуку. У двійковому дереві пошуку пошук починається з кореня і щоразу приймається двостороннє рішення (піти по лівому піддереву або правому). У В-дереві пошук також починається з кореневого вузла, але на кожному кроці приймається n-стороннє рішення, де n - це загальна кількість нащадків вузла, що розглядається. У дереві складність пошуку становить O(log n). Пошук відбувається так:

Крок 1: Отримати елемент для пошуку.

Крок 2: Порівняти елемент з першим значенням ключа в кореневому вузлі дерева.

Крок 3: Якщо вони збігаються, вивести: «Значення знайдене!» та завершити пошук.

Крок 4: Якщо вони не збігаються, перевірте чи більше або менше значення елемента, ніж поточне значення ключа.

Крок 5: Якщо потрібний елемент менший, продовжити пошук по лівому піддереву.

Крок 6: Якщо потрібний елемент більший, порівняти елемент з наступним значенням ключа у вузлі і повторювати Кроки 3, 4, 5 і 6 доки не знайдено збіг або поки потрібний елемент не буде порівняний з останнім значенням ключа у вузлі-аркуші.

Крок 7: Якщо останнє значення ключа у вузлі-листі не збіглося з шуканим, вивести, що значення не знайдено.

***Операція вставки в B-дерево:***

У дереві новий елемент може бути доданий тільки в вузол-лист. Це означає, що нова пара ключ-значення завжди додається тільки до вузла-лист. Вставка відбувається так:

Крок 1: Перевірити чи порожнє дерево.

Крок 2: Якщо дерево порожнє, створити новий вузол з новим значенням ключа і прийняти його за кореневий вузол.

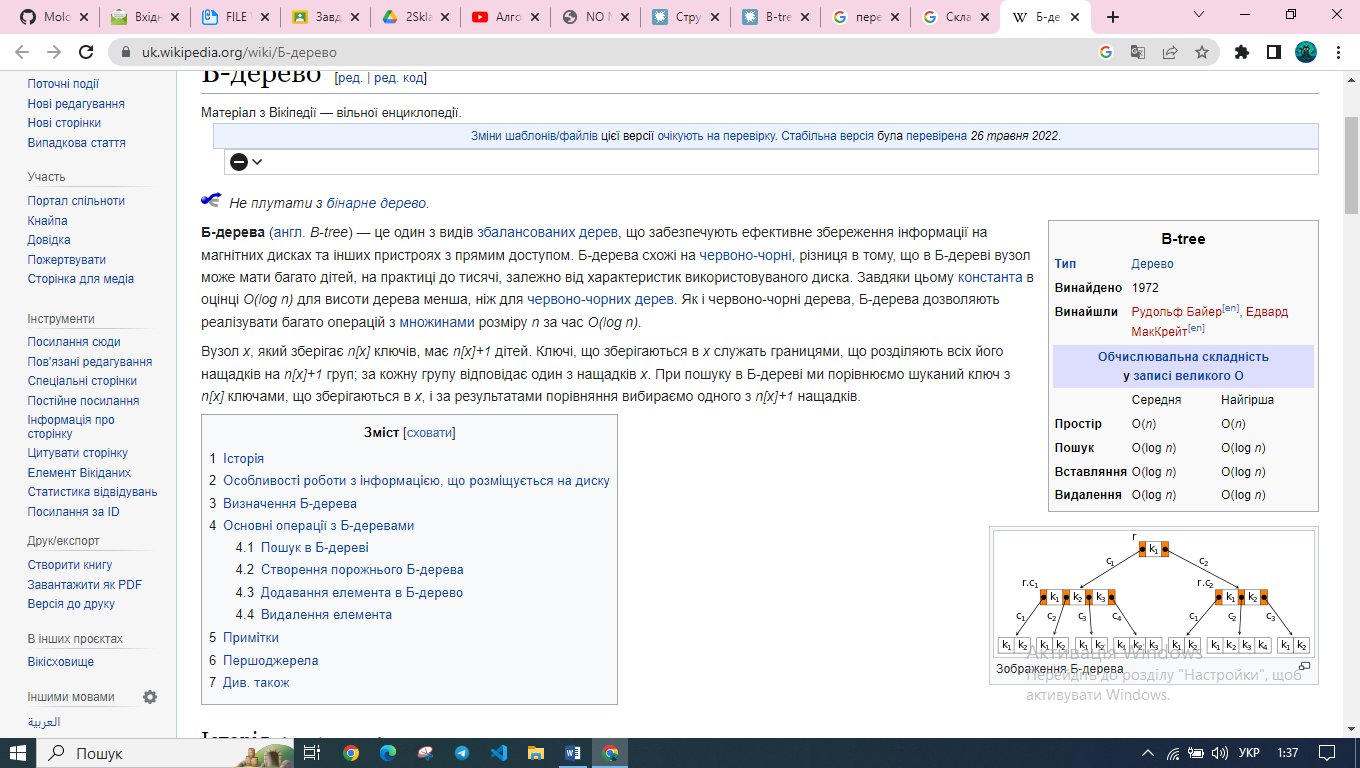
Крок 3: Якщо дерево не порожнє, знайти відповідний вузол-лист, до якого буде додано нове значення, використовуючи логіку дерева двійкового пошуку.

Крок 4: Якщо в поточному вузлі-листі є незайнята комірка, додати новий ключ-значення до поточного вузла-листа, слідуючи зростаючому порядку значень ключів усередині вузла.

Крок 5: Якщо поточний вузол повний і не має вільних осередків, розділіть вузол-лист, відправивши середнє батьківському вузлу. Повторюйте крок, доки значення, що надсилається, не буде зафіксовано у вузлі.

Крок 6: Якщо поділ відбувається з коренем дерева, тоді середнє значення стає новим коренем дерева та висота дерева збільшується на одиницю.

**Складності**



Якщо порахувати кількість ітерацій в моїй реалізації, то значення близькі до наведених в таблиці(близькі, а не збігаються, скоріш за все через оцінку, константи в оцінці відкидаються).

**Мова програмування**

Мова програмування: С++;

Середовище розробки: Visual Studio 2022;

Додаткові інструменти: GitHub;

Посилання на репозиторій GitHub: <https://github.com/MoloZzz/B-tree>

**Модулі програми**

**Модулі для роботи з даними:**

Класи:

class Complex //клас для задання й роботи з комплексними числами

Приватні змінні класу:

double re, im; //re - дійсна частина, im - коефіцієнт біля уявної частини

Конструктори:

Complex(); //порожній конструктор(на всякий випадок)

Complex(double r); //конструктор для змінної що не містить уявної частини

Complex(double r, double i); //стандартний конструктор

Complex(const Complex& c); //конструктор копіювання

Функції класу:

double abs(); // Модуль комплексного числа

string getString(); // Функція для переведення числа в строку(для виведення в консоль)

Оператори:

Complex& operator = (Complex& c); // оператор присвоювання( присвоює значення першої змінної другій),приклад в коді:

Complex k = Complex(10, 2); рядок 187 файл B-tree.cpp

Далі три оператори порівняння(використовуються для визначення ,яке значення більше)

bool operator < (Complex c);

bool operator > (Complex c);

bool operator == (Complex c);

**Модулі самого дерева:**

Класи:

class TreeNode; // Клас для функцій і параметрів вузла

class BTree; // Клас для функцій і параметрів дерева

Параметри класу TreeNode:

Complex\* keys; // масив значень вузла(впорядкований)

int t; // порядок дерева (якщо вибрали, що максимальна кількість значень у вузлі m,то порядок дерева m + 1)

TreeNode\*\* C; // масив посилань на листя(синів)

int n; // кількість значень у вузлі

bool leaf; // перевірка чи є листки(сини) у вузла

Функції та конструктори класу TreeNode:

TreeNode(int temp, bool bool\_leaf); // базовий конструктор(temp – задає порядок,bool\_leaf – показує чи є листя)

void insertNonFull(Complex k); // вставка значення у вузол(поки він не має максимуму значень), k - значення

void splitChild(int i, TreeNode\* y); // функція розбиття, і – кількість значень у вузлі, у – посилання,аби перевірити чи є листя у вузла

void traverse(); // функція проходу по значеннях вузла і дерева(виконана рекурсивно)

TreeNode\* search(Complex k); // функція пошуку значення, к - значення

friend class BTree; // надання доступу до полів класу BTree

Параметри класу BTree:

TreeNode\* root; // посилання на корінь

int t; // порядок дерева

Функції та конструктори класу BTree:

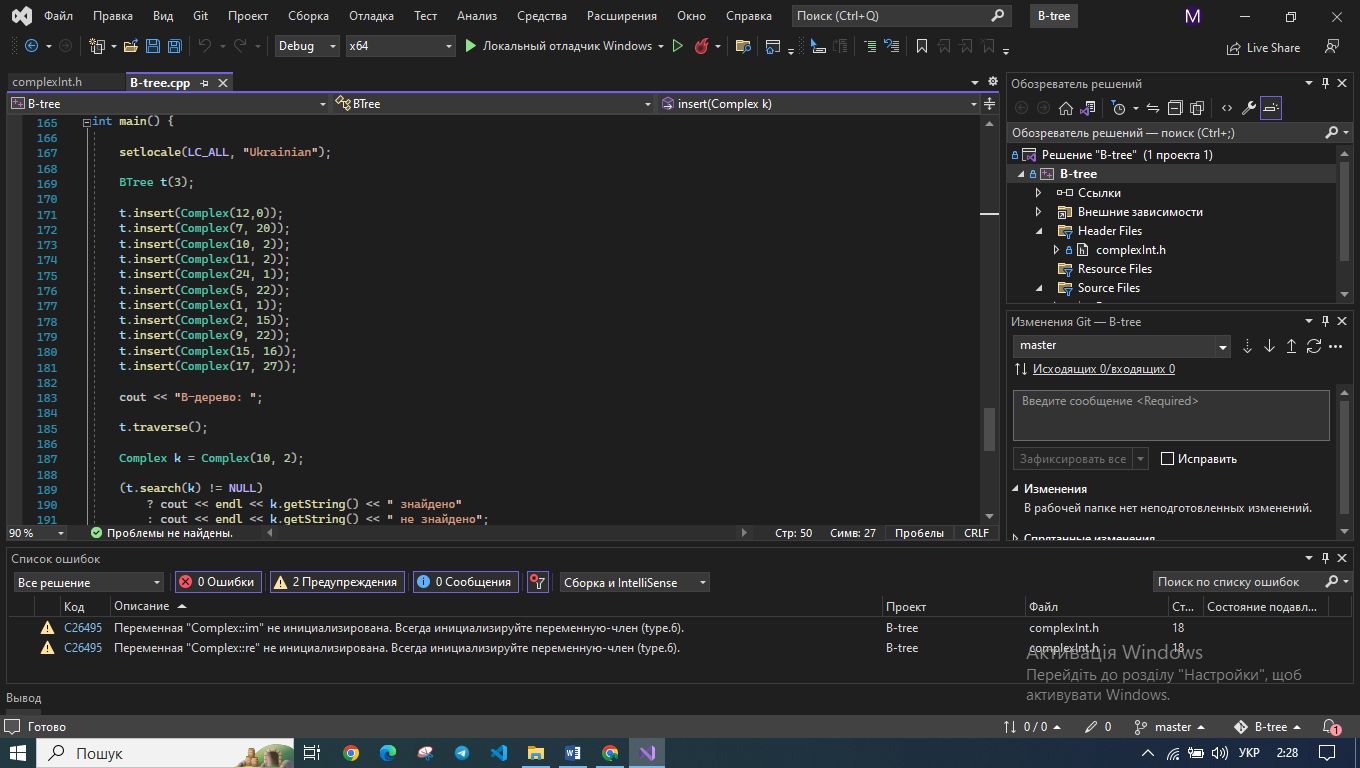
BTree(int temp) // базовий конструктор(корінь пустий, temp – задає порядок дерева)

void traverse() // функція проходу по значеннях вузла і дерева(виконана рекурсивно)

TreeNode\* search(Complex k) // функція пошуку значення

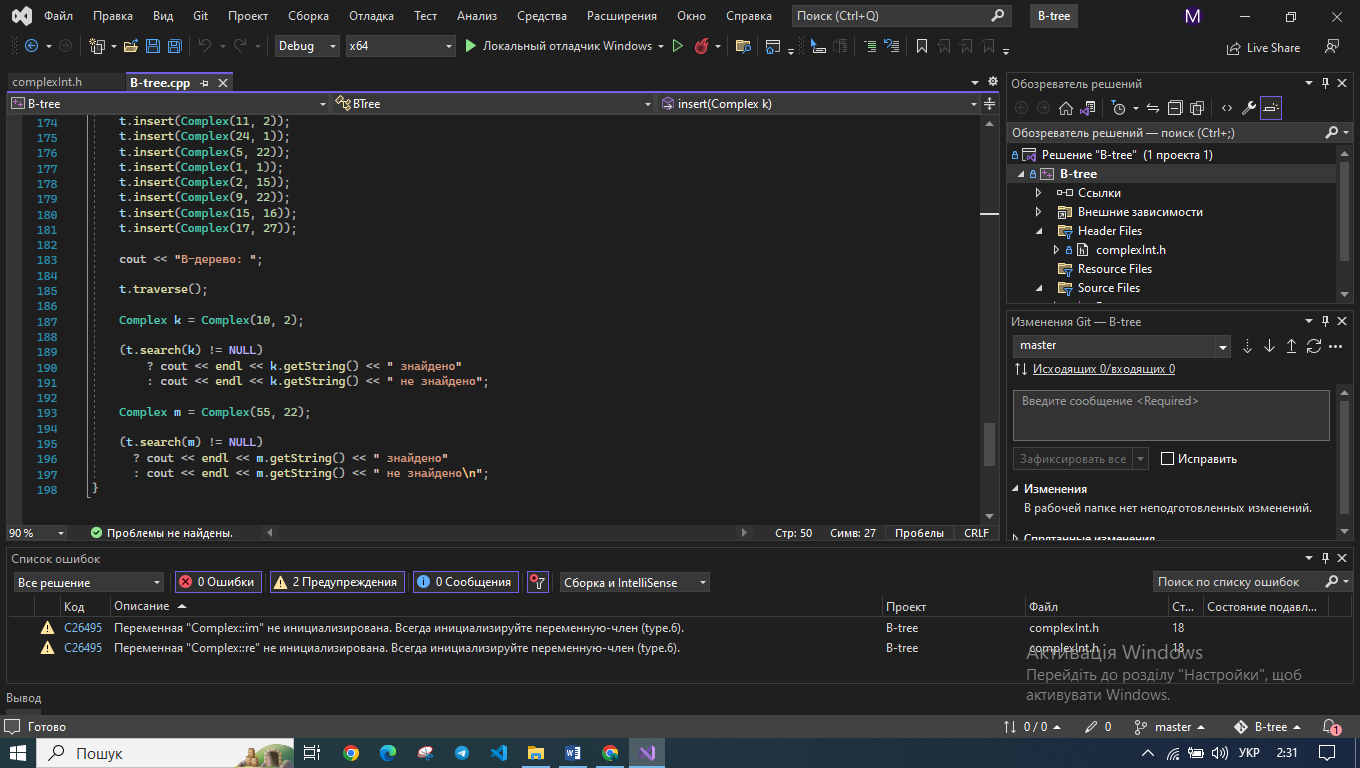
void insert(Complex k) // функція вставки, к - значення, що додають.

**Інтерфейс користувача**



При заданні класу, в параметр ми вказуємо порядок дерева.

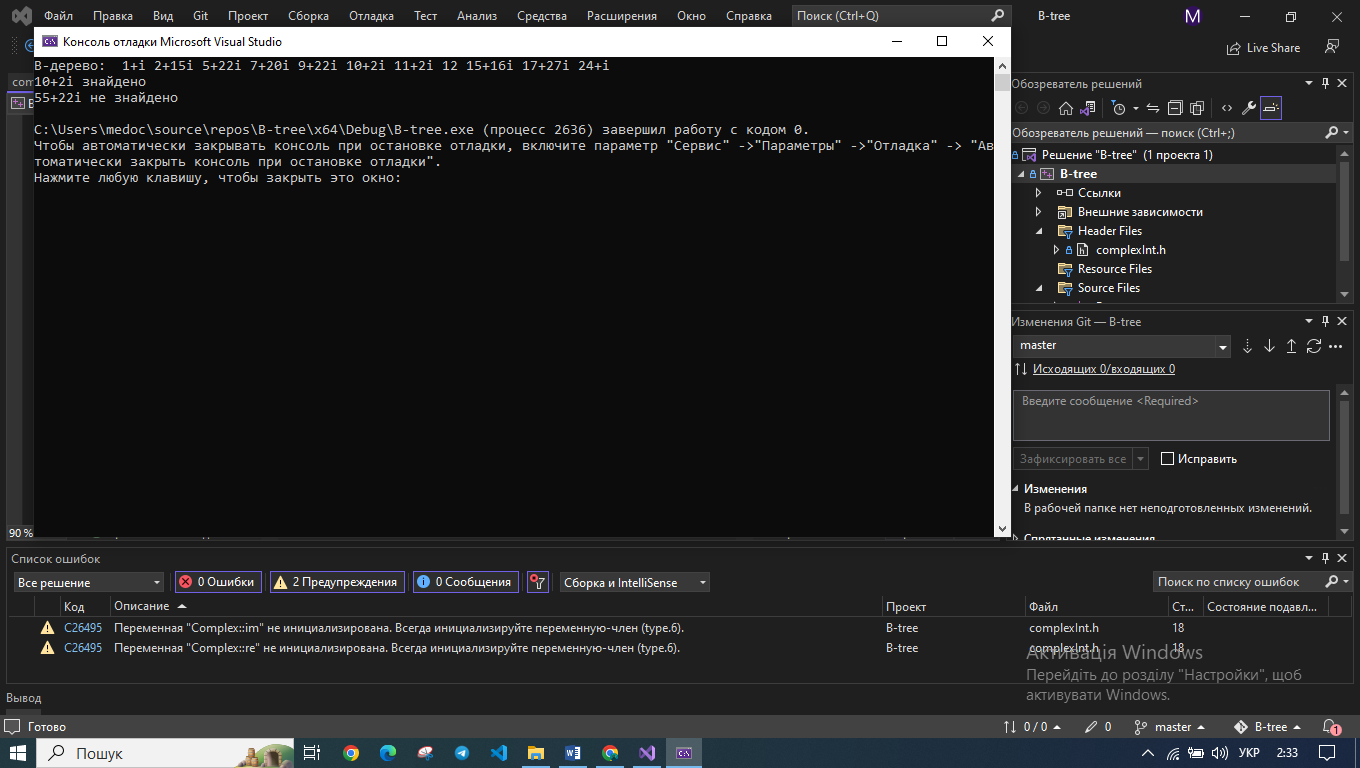
Додавати значення ми можемо за допомогою взаємодії з класом,функція insert.



Для виведення дерева ми використовуємо функцію traverse,

Для пошуку використовуємо функцію search().

Результат запитів:



**Джерела**

Лекції «Алгоритми і Складність» О. Шкільняк

<https://uk.wikipedia.org/wiki/%D0%91-%D0%B4%D0%B5%D1%80%D0%B5%D0%B2%D0%BE>

<https://habr.com/ru/companies/otus/articles/459216/>

<https://habr.com/ru/articles/114154/>

<https://codechick.io/tutorials/dsa/dsa-b-tree>

https://uk.myservername.com/b-tree-b-tree-data-structure-c