**Міністерство освіти і науки, МОЛОДІ І СПОРТУ України**

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ

імені ІГОРЯ СІКОРСЬКОГО”

**Кафедра конструювання ЕОА**

**ЗВІТ**

з лабораторної роботи №3  
по курсу «Алгоритмічні мови та програмування – 2»  
на тему «Дерева»

Виконав:

студент гр. ДК-12

Полов’ян Д.В.

Дата виконання: 02.05.2022

Перевірив:

ст. викладач Губар В.Г.

Київ – 2022**Лабораторна робота №2**

***Тема роботи: Зв’язані списки.***

***Тема роботи: Бінарні дерева.***

Мета роботи: Складання алгоритмів з використанням бінарних дерев.

Завдання: створити бінарне дерево (виберіть згідно Вашого завдання), вивести його. Реалізувати функції: друку дерева (тип обходу згідно варіанту), створення/видалення дерева, вставки елементу, видалення елементу, пошуку елементу.

**2-й варіант** - дерево містить елементи типу int, тип обходу – прямий.

main.c

intTree \*LAB3 = createIntTree()

insertIntValueToTree

(один й більше)

printTree(LAB3->root)

(друкуємо дерево)

findNodeWithValue(LAB3, 2)->value)

(перевіряємо на існування)

printTree(LAB3->root)

(друкуємо дерево)

deleteIntTree(LAB3);

(видаляємо дерево)

findNodeWithValue(LAB3, 2)->value)

(перевіряємо на існування)

deleteValueFromTree(LAB3, 97)

(один й більше)

createNodeWithIntValue()

виділяємо пам’ять для елемента

створюємо наш елемент

присвоюємо NULL лівому нащадку

присвоюємо значення NULL лівому і правому попередникам

перевіряємо дерево на існування

о дерево на існування

кінець функції

існує

не існує

createIntTree()

виділяємо пам’ять для дерева

повертаємо значення дерева

перевіряємо дерево на існування

о дерево на існування

присвоюємо початкове значення NULL

присвоюємо значення count = 0

кінець функції

не існує

існує

deleteNode()

перевіряємо чи існує елемент

видаляємо лівого і правого попередників

видаляємо елемент

кінець функції

не існує

існує

deleteValueFromTree()

в елемента є нащадки

видаляємо елемент

повертаємо результат

дадаємо елемент , який має бути більше лівого нащадка і не менше за правого і ставимо його на місце вибраного елемента

в елемента один нащадок

ставимо нащадок на місце видаленого елемента

ні

ні

так

так

кінець функції

insertIntValueToTree()

перевіряємо дерево на існування

о дерево на існування

створюємо наш елемент

перевіряємо на на існування

кінець функції

перевіряємо на буття коренем

вставляємо елемент

збільшуємо розмір дерева на одиницю

існує

існує

не існує

не існує

корінь

не корінь

кінець функції

**Висновок**

Протягом цієї лабораторної роботи ми дізналися що воно таке і дослідили бінарне дерево: навчилися його обходити, додавати елементи, шукати елементи, правильно видаляти елементи, які мають нащадків (вставляючи замість них число яке по правилам бінарного дерева може знаходитись на тому місці). У нашому варіанті ми використовували і засвоїли прямий обхід бінарного дерева: принцип полягає у тому що ми проходимо до лівого крайнього вузла і разом з цим відвідуємо кожен вузол. Коли у лівій частині дерева закінчились елементи ми переходимо до правої частини дерева і знову його проходимо, разом з відвідуванням усіх вузлів.