**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-11

Осьмірко Є. О.

Перевірив:

ст. викладач

Губар В. Г.

Київ – 2022

**Лабораторна робота №2**

**Списки**

**Тема роботи:** Бінарні дерева.

**Мета роботи:** Складання алгоритмів з використанням бінарних дерев.

**Завдання 12.**

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.

Дерево містить елементи типу char, тип обходу – зворотній.

**Висновок:** При написанні програми я ознайомився з складанням алгоритмів з використанням бінарних дерев. Ціль цих дерев це: швидкі операції вставки/видалення та швидкого пошуку. Дізналися, що є 3 види обходу дерева: прямий, симетричний та зворотній.

main.c

TreePtr tree = createTree()

insertNode(value)

(додавання значень)

printTree(tree)

(друкуємо дерево)

findNode(tree, “abc”)

->value

(перевіряємо на існування)

printTree(tree)

(друкуємо дерево)

destroyTree(tree);

(видаляємо дерево)

deleteNode(tree, a)

(видалення елементу)

insertNode()

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

createTree()

виділяємо пам’ять для дерева

повертаємо значення дерева

перевіряємо дерево на існування

о дерево на існування

присвоюємо початкове значення NULL

кінець функції

не існує

існує

destroyNode()

перевіряємо чи існує елемент

видаляємо лівого і правого попередників

видаляємо елемент

кінець функції

не існує

існує

deleteNode()

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