**Міністерство освіти і науки України**

**Національний технічний університет України «Київський політехнічний**

**інститут імені Ігоря Сікорського"**

**Факультет інформатики та обчислювальної техніки**

**Кафедра інформатики та програмної інженерії**

**Звіт**

з лабораторної роботи № 6 з дисципліни

«Алгоритми та структури даних 2. Структури даних»

**«Деревовидні структури даних»**

**Виконав** ІП-11Трикош Іван Володимирович

**Перевірила** Халус Олена Андріївна

Київ 2022

1 МЕТА ЛАБОРАТОРНОЇ РОБОТИ

Мета роботи – вивчити основні підходи формалізації та імплементації

алгоритмів побудови та обробки базових деревовидних структур даних.

2 ЗАВДАННЯ

Розробити алгоритм розв’язання задачі відповідно до варіанту. Виконати програмну реалізацію задачі. Не використовувати вбудовані деревовидні структури даних (контейнери). Зробити висновок по лабораторній роботі.

31. Побудувати дерево, елементами якого є цілі числа. Видалити з дерева усі листки, надрукувати початкове та модифіковане дерево.

3 ВИКОНАННЯ

3.1 Псевдокод алгоритму

class Tree:

value – значення елемента

left, right – ліва та права гілка

Tree(value1):

value = value1

left = None

right = None

end Tree

method output\_tree():

if left != None:

left.output\_tree()

end if

if value != None:

print(value)

end if

if right != None:

right.output\_tree()

end if

end method output\_tree

method delete():

flag = True

if left != None:

left.delete()

flag = False

end if

if right != None:

right.delete()

flag = False

end if

if left == None and right == None and flag:

value = None

end if

end method delete

function input\_tree():

value = int(input("Введіть елемент дерева (Ціле число): "))

tree = Tree(value)

n = int(input("Введіть кількість нащадків (0, 1, 2): "))

if n == 2:

tree.left = input\_tree()

tree.right = input\_tree()

elif n == 1:

tree.left = input\_tree()

return tree

end function input\_tree

3.2 Програмна реалізація алгоритму

3.2.1 Вихідний код

class Tree():

def \_\_init\_\_(self, value):

self.value = value # Елемент

self.left = None # Ліва гілка

self.right = None # Права гілка

# Обходимо центровано дерево та виводимо його

def output\_tree(self):

if self.left != None:

self.left.output\_tree()

if self.value != None:

print(self.value, end = " ")

if self.right != None:

self.right.output\_tree()

# Видаляємо усі листки (ті елементи, у яких ліва та права гілка = None)

def delete(self):

if self.left != None:

self.left.delete() # Обходимо ліву гілку

if self.right != None:

self.right.delete() # Обходимо праву гілку

if self.left == None and self.right == None: # Видаляємо листок

self.value = None

def input\_tree():

tree = Tree(int(input("Введіть елемент дерева (Ціле число): "))) # Створюємо гілку дерева

n = int(input("Введіть кількість нащадків (0, 1, 2): ")) # Вводимо кількість нащадків

if n == 2:

tree.left = input\_tree() # Вводимо ліву гілку

tree.right = input\_tree() # Вводимо праву гілку

elif n == 1:

tree.left = input\_tree() # Вводимо ліву гілку

return tree # Повертаємо гілку

def main():

tree = input\_tree() # Вводимо дерево

print("Початкове дерево: ")

tree.output\_tree() # Виводимо початкове дерево

print("\nМодифіковане дерево: ")

tree.delete() # Видаляємо листки

tree.output\_tree() # Виводимо модифіковане дерево

print()

if \_\_name\_\_ == "\_\_main\_\_":

main()

3.2.2 Приклад роботи

На рисунках 3.1 і 3.2 показані приклади роботи програми.

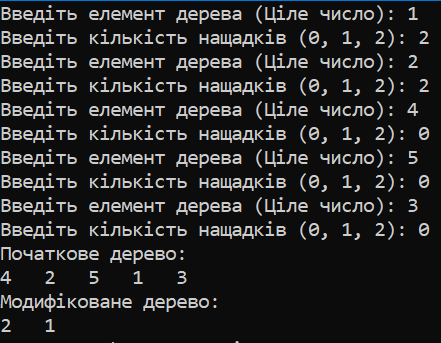


Рисунок 3.1

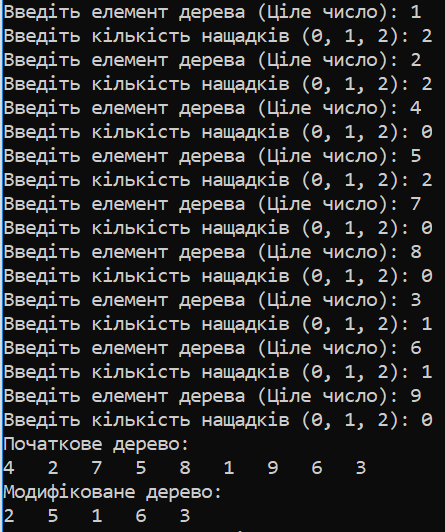


Рисунок 3.2

ВИСНОВОК

При виконанні даної лабораторної роботи я вивчив основні підходи формалізації та імплементації алгоритмів побудови та обробки базових деревовидних структур даних; побудував дерево, елементами якого є цілі числа та видалив з нього усі листки; навчився обходити дерево в прямому, зворотному та центрованому порядку; набув навичок роботи з деревами на мові Python.