Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего образования

«Владимирский государственный университет

имени Александра Григорьевича и Николая Григорьевича Столетовых»

(ВлГУ)

Колледж инновационных технологий и предпринимательства

**Лабораторная работа №1**

на тему: «Деревья»

Вариант 21

Выполнила:

Ст.гр. ИРсп-121

Орлова. С. В

Проверил:

Ларин Е. С.

Владимир 2023

**Цель работы**

Изучить способы эффективного хранения и обработки информации на примере бинарных деревьев.

**Ход работы**

**Задание**

Написать функцию, которая определяет максимальный элемент из всех листьев непустого бинарного дерева.

**Программный код**

from random import randint

from PIL import Image, ImageDraw, ImageFont

class Node:

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

self.data = value

self.left = None

self.right = None

self.id = name

class BinaryTree:

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

self.root = None

self.count = 0

# region Public

def insert(self, val: int, name: int):

if self.root is not None:

self.\_insert(val, self.root, name)

else:

self.root = Node(val, name)

self.count += 1

def drawTree(self):

img = Image.new("RGB", (1000, 1000), "white")

draw = ImageDraw.Draw(img)

font = ImageFont.truetype("arial.ttf",20)

if self.root is not None:

self.\_drawTree(self.root,500,25,250,50,draw,font)

return img

def delete\_tree(self,node: Node):

if node is not None :

self.delete\_tree(node.left)

self.delete\_tree(node.right)

del node

def dfs\_viewing(self, node: Node):

if node:

self.dfs\_viewing(node.left)

print(node.data, end=' ')

self.dfs\_viewing(node.right)

def dfs(self, node: Node):

if node:

steck = self.dfs(node.left)

steck.append(node.data)

steck += self.dfs(node.right)

return steck

else:

return []

# region Private

def \_insert(self, value: int, node: Node, name: int):

if value < node.data:

if node.left is not None:

self.\_insert(value, node.left, name)

else:

node.left = Node(value,name)

else:

if node.right is not None:

self.\_insert(value, node.right, name)

else:

node.right = Node(value,name)

def \_drawTree(self,node,x,y,dx,dy,draw,font):

if node.left is not None :

newx,newy = x-dx,y+dy

draw.line([(x,y),(newx,newy)],width=2,fill='black')

self.\_drawTree(node.left,newx,newy,dx//2,dy,draw,font)

if node.right is not None:

newx,newy = x+dx,y+dy

draw.line([(x,y),(newx,newy)],width=2,fill='black')

self.\_drawTree(node.right,newx,newy,dx//2,dy,draw,font)

draw.ellipse( (x-25,y-25,x+25,y+25),fill='white', outline='black' )

if node.data < 10 :

draw.text((x-7,y-7),str(node.data),(0,0,0),font)

else:

draw.text((x-10,y-7),str(node.data),(0,0,0),font)

def find\_maximum(self,node):

if node is None:

return float('-inf')

if node.left is None and node.right is None:

return node.data

left\_max = self.find\_maximum(node.left)

right\_max = self.find\_maximum(node.right)

return max(left\_max, right\_max)

def main():

tree = BinaryTree()

for i in range(10):

tree.insert(randint(1,10), i)

img1 = tree.drawTree()

img1.show()

dfs\_steck = tree.dfs(tree.root)

print(dfs\_steck)

max\_element = tree.find\_maximum(tree.root)

print("max element:", max\_element)

tree.delete\_tree(tree.root)

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

main()

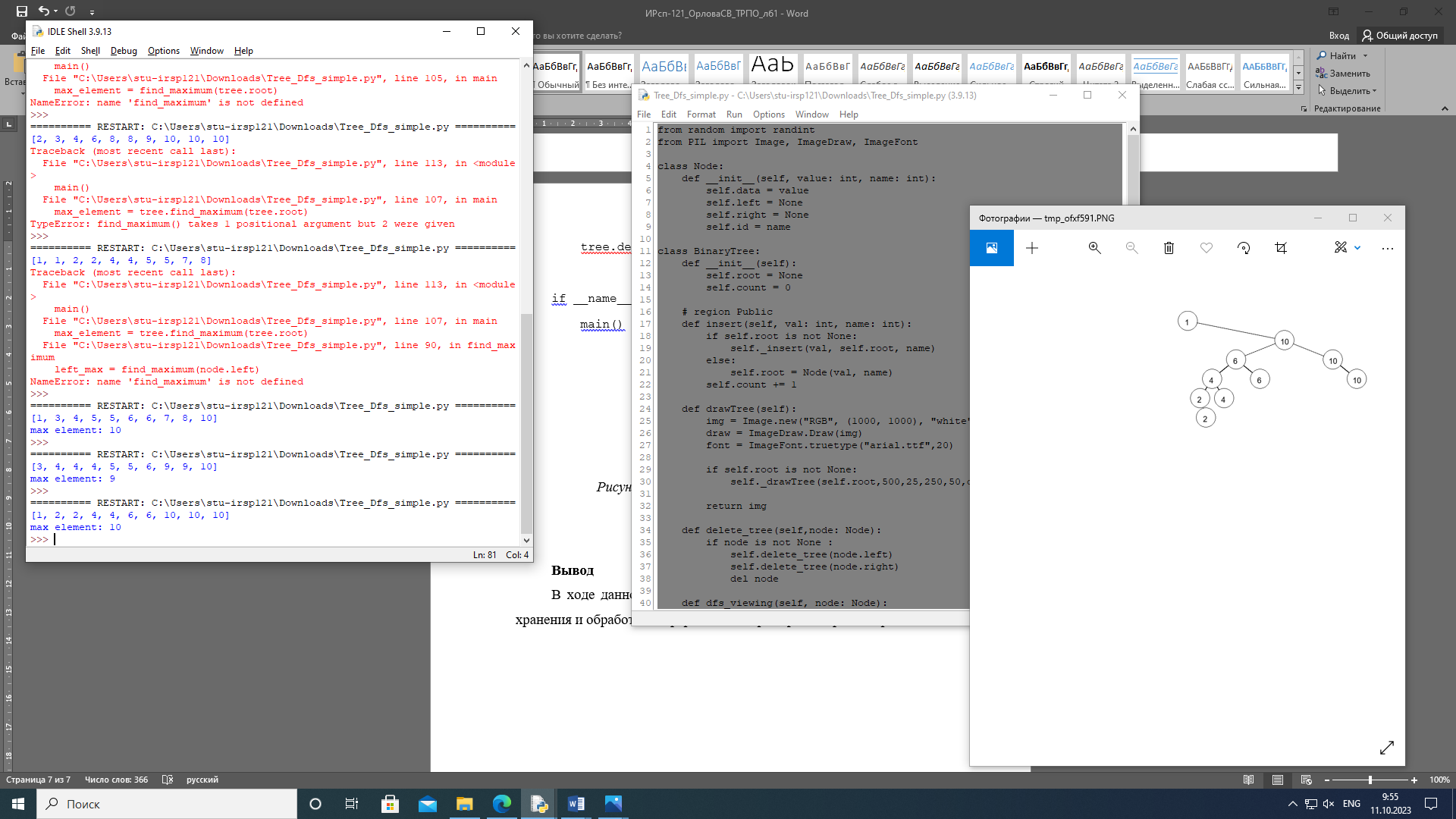
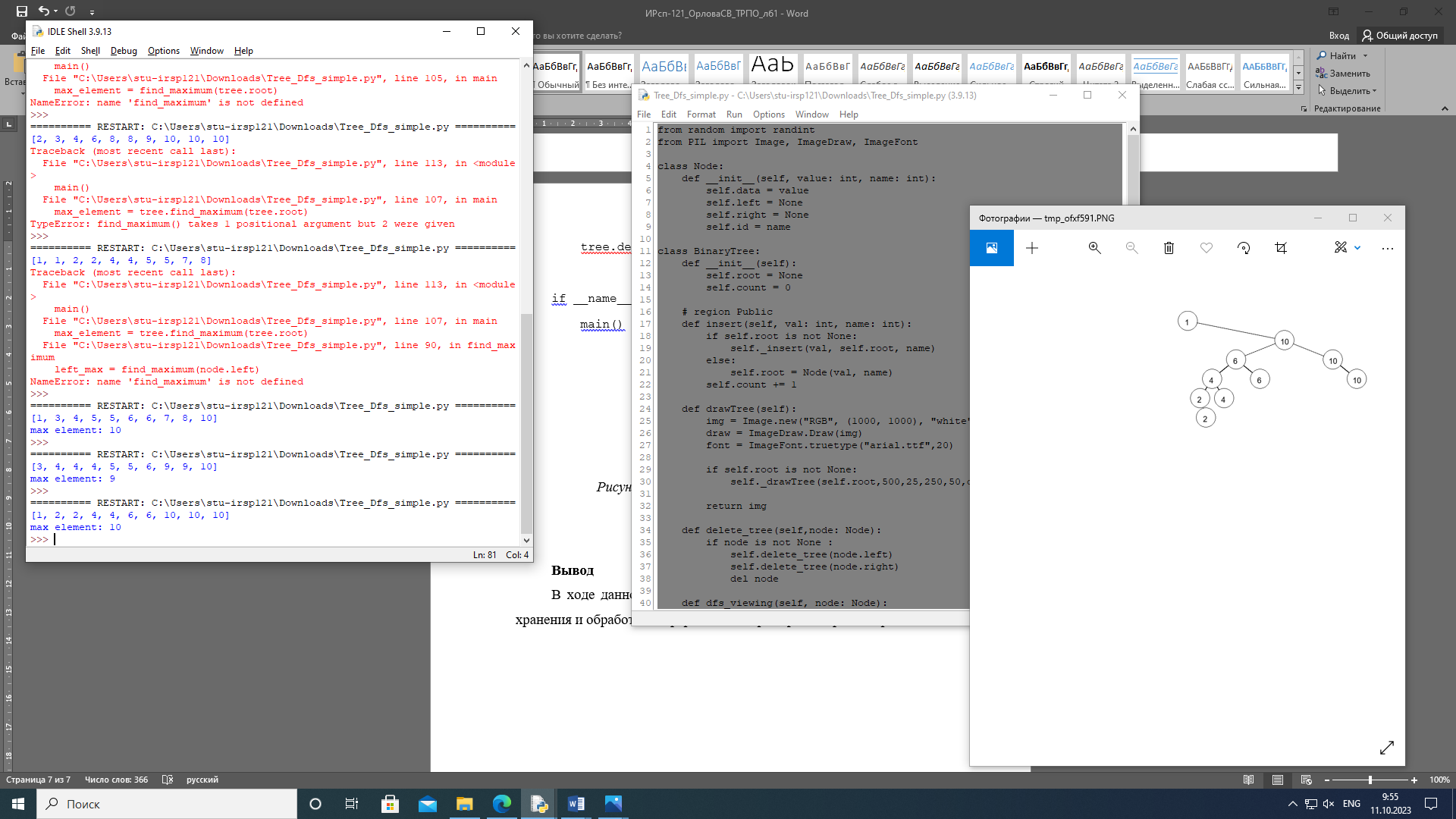


Рисунок 1 - Результат работы программного кода



**Вывод**

В ходе данной лабораторной работы изучены способы эффективного хранения и обработки информации на примере бинарных деревьев.