МІНІСТЕРСТВО ОСВІТИ І НАУКИ
ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ РАДІОЕЛЕКТРОНІКИ

ЗВІТ

з виконання лабораторної роботи №3

з дисципліни «Функціональне програмування»

за темою «Функції вищого порядку. Знайомство з можливостями функціонального програмування мови Python»

Виконав:
Ковалик Вадим Валерійович

Перевірив:
Міхаль Олег Пилипович

Харків 2024

3.1 Мета роботи

Практичне відпрацювання прийомів використання елементів і підходів функціонального програмування у складі та в комбінації з процедурними імперативними складовими програми.

3.2 Порядок виконання роботи

3.2.1 Завдання:

На даній лабораторній роботі ми повинні запустити програму, перевірити її працездатність та переконатися, що вона видає протокол. Після цього слід дописати програму у процедурному стилі. Завершальним кроком є переписання програми з використанням функціонального стилю.

3.2.2 Хід роботи

3.2.2.1 Перевірка працездатності заданої програми

Запускаємо програму та перевіряємо її працездатність та переконуємося, що вона видає протокол.

Лістинг 3.1 – Код програми для перевірки працездатності

```
import random
Protocol_name = "Protocol_name.txt"
file = open(Protocol_name, "a")
file.write("======== Beginning of the protocol " + Protocol_name + " ========\n")
print("======== Beginning of the protocol " + Protocol_name + " ========")
nn = 20
file.write("nn = " + str(nn) + "\n")
print("nn = " + str(nn))
Q = [0] * nn
file.write("Q (initial) = " + str(Q) + "\n")
print("Q (initial) = " + str(Q))
for i in range(0, nn):
    Q[i] = int(random.random() * 255)
file.write("Q (filled) = " + str(Q) + "\n")
print("Q (filled) = " + str(Q))
MaxQ = max(Q)
MinQ = min(Q)
file.write("MaxQ = " + str(MaxQ) + ", MinQ = " + str(MinQ) + "\n")
print("MaxQ = " + str(MaxQ) + ", MinQ = " + str(MinQ))
file.write("======== End of protocol " + Protocol_name + " ========\n")
print("======== End of protocol " + Protocol_name + " ========")
file.close()
print("Protocol is generated")
z = 0
while z != "":
    print("=======================================")
    z = input("Press ENTER for exit ... ")
```

```
======== Beginning of the protocol Protocol_name.txt ========
nn = 20
Q (initial) = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
Q (filled) = [96, 88, 87, 9, 172, 96, 171, 151, 13, 191, 181, 61, 37, 74, 25, 78, 54, 108, 197, 86]
MaxQ = 197, MinQ = 9
======== End of protocol Protocol_name.txt ========
Protocol is generated
=====================================
Press ENTER for exit ...
```

Рисунок 3.1 – Результат виконання в консолі

```
======== Beginning of the protocol Protocol_name.txt ========
nn = 20
Q (initial) = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
Q (filled) = [96, 88, 87, 9, 172, 96, 171, 151, 13, 191, 181, 61, 37, 74, 25, 78, 54, 108, 197, 86]
MaxQ = 197, MinQ = 9
======== End of protocol Protocol_name.txt ========
```

Рисунок 3.2 – Вигляд файлу Protocol_name

3.2.2.2 Доповнення програми в процедурному стилі

Розширюємо програму, використовуючи процедурний стиль, дотримуючись коментаря, що міститься в її заключній частині.

Лістинг 3.2 – Доповнена програма згідно коментаря

```
import random

Protocol_name = "Protocol_name.txt"
file = open(Protocol_name, "a")
file.write("======== Beginning of the protocol " + Protocol_name + " ========\n")
print("======== Beginning of the protocol " + Protocol_name + " ========")
nn = 20
file.write("nn = " + str(nn) + "\n")
print("nn = " + str(nn))
Q = [0] * nn
file.write("Q (initial) = " + str(Q) + "\n")
print("Q (initial) = " + str(Q))
for i in range(0, nn):
    Q[i] = int(random.random() * 255)
file.write("Q (filled) = " + str(Q) + "\n")
print("Q (filled) = " + str(Q))
MaxQ = max(Q)
MinQ = min(Q)
file.write("MaxQ = " + str(MaxQ) + ", MinQ = " + str(MinQ) + "\n")
print("MaxQ = " + str(MaxQ) + ", MinQ = " + str(MinQ))
range_step = (MaxQ - MinQ) / 5
histogram = [0] * 5
for value in Q:
    index = min(int((value - MinQ) / range_step), 4)
    histogram[index] += 1
file.write("Histogram = " + str(histogram) + "\n")
print("Histogram = " + str(histogram))
file.write("======== End of protocol " + Protocol_name + " ========\n")
print("======== End of the protocol " + Protocol_name + " ========")
```

```
file.close()
print("Protocol is generated")
z = 0
while z != "":
    print("========================================")
    z = input("Press ENTER for exit ... ")
```

```
======== Beginning of the protocol Protocol_name.txt ========
nn = 20
Q (initial) = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
Q (filled) = [98, 97, 115, 82, 125, 11, 236, 229, 130, 135, 56, 53, 159, 59, 165, 146, 15, 161, 72, 34]
MaxQ = 236, MinQ = 11
Histogram = [4, 6, 4, 4, 2]
======== End of the protocol Protocol_name.txt ========
Protocol is generated
====================================
Press ENTER for exit ...
```

Рисунок 3.3 – Результат виконання розширеної програми

```
======== Beginning of the protocol Protocol_name.txt ========
nn = 20
Q (initial) = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
Q (filled) = [98, 97, 115, 82, 125, 11, 236, 229, 130, 135, 56, 53, 159, 59, 165, 146, 15, 161, 72, 34]
MaxQ = 236, MinQ = 11
Histogram = [4, 6, 4, 4, 2]
======== End of protocol Protocol_name.txt ========
```

Рисунок 3.4 – Вигляд файлу Protocol_name

3.2.2.3 Переписання програми у функціональному стилі

Переписуємо програму з максимальним використанням функціонального стилю відповідно до принципів.

Лістинг 3.3 – Використання функціонального стилю

```
import random

def initialize_protocol_file(filename):
    file = open(filename, "a")
    file.write("======== Beginning of the protocol " + filename + " ========\n")
    print("======== Beginning of the protocol " + filename + " ========")
    return file

def close_protocol_file(file, filename):
    file.write("======== End of protocol " + filename + " ========\n")
    print("======== End of protocol " + filename + " ========")
    file.close()

def generate_random_array(size, max_value=255):
    return [int(random.random() * max_value) for _ in range(size)]

def calculate_histogram_bounds(min_val, max_val, num_bins=5):
    range_step = (max_val - min_val) / num_bins
    return [min_val + i * range_step for i in range(1, num_bins)]
```

```
def populate_histogram(array, bounds):
    histogram = [0] * (len(bounds) + 1)
    for value in array:
        for i, bound in enumerate(bounds):
            if value < bound:
                histogram[i] += 1
                break
        else:
            histogram[-1] += 1
    return histogram

def main_protocol(filename="Protocol_name.txt", array_size=20):
    file = initialize_protocol_file(filename)
    Q = generate_random_array(array_size)
    file.write("Q = " + str(Q) + "\n")
    print("Q = " + str(Q))
    MaxQ = max(Q)
    MinQ = min(Q)
    file.write("MaxQ = " + str(MaxQ) + ", MinQ = " + str(MinQ) + "\n")
    print("MaxQ = " + str(MaxQ) + ", MinQ = " + str(MinQ))
    bounds = calculate_histogram_bounds(MinQ, MaxQ)
    histogram = populate_histogram(Q, bounds)
    file.write("Histogram = " + str(histogram) + "\n")
    print("Histogram = " + str(histogram))
    close_protocol_file(file, filename)
    print("Protocol is generated")

main_protocol()
```

```
======== Beginning of the protocol Protocol_name.txt ========
Q = [201, 174, 100, 52, 99, 82, 178, 147, 9, 91, 78, 16, 90, 6, 7, 220, 213, 46, 131, 202]
MaxQ = 220, MinQ = 6
Histogram = [5, 5, 3, 2, 5]
======== End of protocol Protocol_name.txt ========
Protocol is generated
```

Рисунок 3.5 – Результат функціональної програми

```
======== Beginning of the protocol Protocol_name.txt ========
Q = [201, 174, 100, 52, 99, 82, 178, 147, 9, 91, 78, 16, 90, 6, 7, 220, 213, 46, 131, 202]
MaxQ = 220, MinQ = 6
Histogram = [5, 5, 3, 2, 5]
======== End of protocol Protocol_name.txt ========
```

Рисунок 3.6 – Вигляд файлу

3.2.2.4 Додавання графічного інтерфейсу за допомогою бібліотеки tkinter

Цей інтерфейс дозволяє користувачеві запускати процес створення протоколу, налаштувати розмір масиву та переглядати результат безпосередньо в програмі.

Лістинг 3.4 – Код створеного графічного інтерфейсу

```
import random
import tkinter as tk
from tkinter import messagebox, scrolledtext


def generate_random_array(size, max_value=255):
    return [int(random.random() * max_value) for _ in range(size)]


def calculate_histogram_bounds(min_val, max_val, num_bins=5):
    range_step = (max_val - min_val) / num_bins
    return [min_val + i * range_step for i in range(1, num_bins)]


def populate_histogram(array, bounds):
    histogram = [0] * (len(bounds) + 1)
    for value in array:
        for i, bound in enumerate(bounds):
            if value < bound:
                histogram[i] += 1
                break
        else:
            histogram[-1] += 1
    return histogram


def generate_protocol(array_size, text_widget):
    Q = generate_random_array(array_size)
    MaxQ = max(Q)
    MinQ = min(Q)
    bounds = calculate_histogram_bounds(MinQ, MaxQ)
    histogram = populate_histogram(Q, bounds)

    protocol_text = (
        f"======== Beginning of the protocol ========\n"
        f"Q = {Q}\n"
        f"MaxQ = {MaxQ}, MinQ = {MinQ}\n"
        f"Histogram = {histogram}\n"
        f"======== End of protocol ========\n"
    )

    text_widget.delete(1.0, tk.END)
    text_widget.insert(tk.END, protocol_text)


def on_generate():
    try:
        array_size = int(entry_size.get())
        if array_size <= 0:
            raise ValueError("The size of the array must be greater than zero.")
        generate_protocol(array_size, text_output)
        messagebox.showinfo("Protocol created", "The protocol is successfully created
and displayed.")
    except ValueError as e:
        messagebox.showerror("Error", str(e))


root = tk.Tk()
root.title("Protocol generator")
frame = tk.Frame(root)
frame.pack(pady=10)
lbl_size = tk.Label(frame, text="The size of the array:")
```

```
lbl_size.pack(side=tk.LEFT)
entry_size = tk.Entry(frame, width=10)
entry_size.pack(side=tk.LEFT)
entry_size.insert(0, "20")
btn_generate = tk.Button(root, text="Create a protocol", command=on_generate)
btn_generate.pack(pady=5)
text_output = scrolledtext.ScrolledText(root, width=50, height=15, wrap=tk.WORD)
text_output.pack(pady=10)
root.mainloop()
```

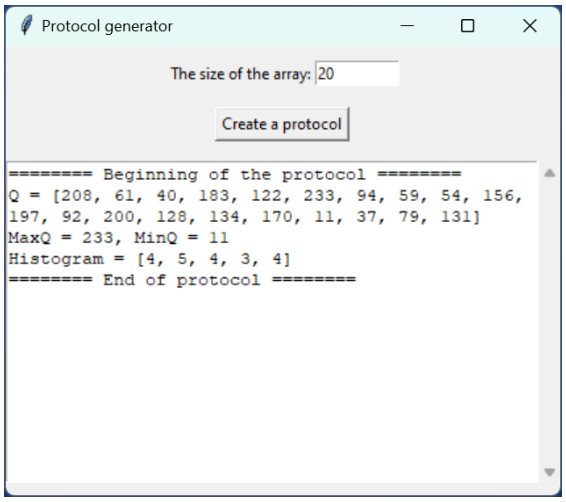

Рисунок 3.7 – Вигляд програми

3.3 Висновок

У даній роботі було реалізовано генерацію протоколу на основі випадкових значень з визначенням мінімальних та максимальних значень, а також побудовою гістограми для візуалізації розподілу даних. Для зручності користувачів створено графічний інтерфейс за допомогою бібліотеки tkinter, що дозволяє налаштовувати параметри масиву та переглядати результати без редагування коду. Це демонструє ефективність автоматизації процесів аналізу даних та можливості, які відкриваються при використанні графічних інтерфейсів у програмуванні.