Звіт

До лабораторної роботи №4

З компютерної графіки

Варіант 1

**Завдання:**

Розробити алгоритм побудови симетрії трикутника відносно довільної прямої L. На екран виводити зображення результатів послідовного виконання кроків алгоритму.

**Код програми**

import matplotlib.pyplot as plt

import numpy as np

import sympy as sym

A = (-2, -4)

B = (5, 4)

C = (5, -1)

L = (-1, 1, 0)  # (a, b, c) - рівняння прямої L: ax + by + c = 0

fig, ax = plt.subplots()

ax.set\_xlim(-10, 10)

ax.set\_ylim(-10, 10)

# Побудова прямої L

x = np.linspace(-10, 10, 100)

y = (-L[2] - L[0] \* x) / L[1]

ax.plot(x, y, 'g-', label='Пряма L')

def reflect\_point(point,line):

    a, b, c = line

    x0, y0 = point

    denominator = a\*\*2 + b\*\*2

    symmetric\_x = x0 - 2 \* (a \* x0 + b \* y0 + c) \* a / denominator

    symmetric\_y = y0 - 2 \* (a \* x0 + b \* y0 + c) \* b / denominator

    return symmetric\_x, symmetric\_y

plt.grid(True)

ax.plot(A[0], A[1], 'ro')

plt.pause(2)

ax.plot(B[0], B[1], 'ro')

plt.pause(2)

ax.plot(C[0], C[1], 'ro')

plt.pause(2)

ax.plot([A[0], B[0], C[0], A[0]], [A[1], B[1], C[1], A[1]], 'b-', label='Початковий трикутник')

plt.xlabel('x')

plt.ylabel('y')

plt.draw()

plt.pause(2)

A\_reflected = reflect\_point(A, L)

ax.plot(A\_reflected[0], A\_reflected[1], 'ro')

plt.pause(2)

B\_reflected = reflect\_point(B, L)

ax.plot(B\_reflected[0], B\_reflected[1], 'ro')

plt.pause(2)

C\_reflected = reflect\_point(C, L)

ax.plot(C\_reflected[0], C\_reflected[1], 'ro')

plt.pause(2)

ax.plot([A\_reflected[0], B\_reflected[0], C\_reflected[0], A\_reflected[0]],

        [A\_reflected[1], B\_reflected[1], C\_reflected[1], A\_reflected[1]], 'r-', label='Симетричний трикутник')

plt.legend()

plt.draw()

plt.pause(2)

plt.show()

**Результат роботи програми**