МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ
УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ
“БРЕСТСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ”
КАФЕДРА ИНТЕЛЛЕКТУАЛЬНЫХ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ

Лабораторная работа №7
По дисциплине “**Современные платформы программирования**”

Выполнил:
студент группы ПО-11
Сильчук Д.А.
Проверил:
Козик И. Д.

Брест 2025

Цель: освоить возможности языка программирования Python в разработке оконных приложений.

Вариант 16

Задание 1. Построение графических примитивов и надписей. Требования к выполнению:
• Реализовать соответствующие классы, указанные в задании;
• Организовать ввод параметров для создания объектов (использовать
экранные компоненты);
• Осуществить визуализацию графических примитивов
Определить класс Rectangle и класс Point. Объявить список из n объектов класса Point. Написать функцию, определяющую, какая из точек лежит снаружи, а какая – внутри прямоугольника.

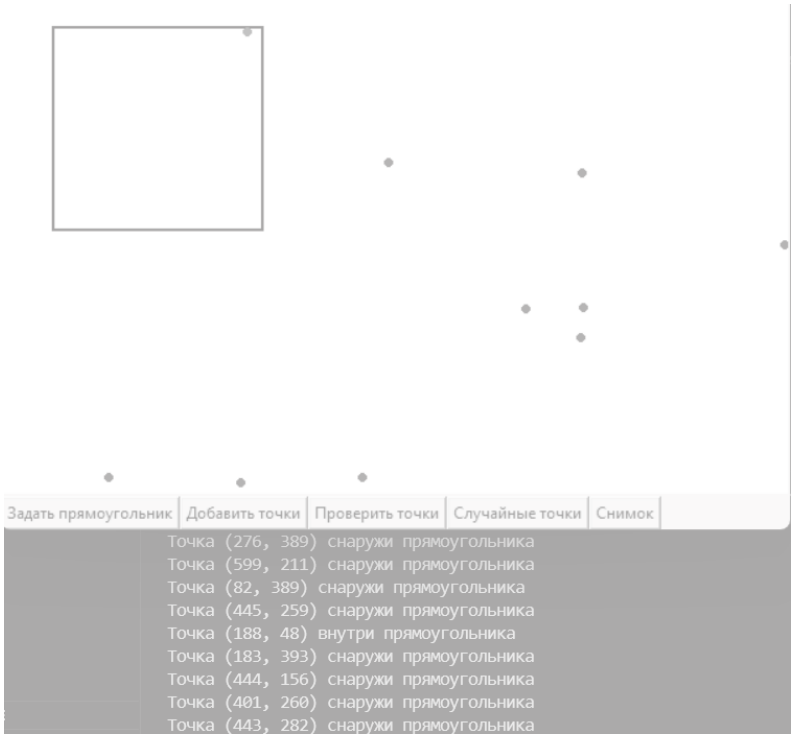
Код программы:

rectangle_points.py

```
import tkinter as tk
from tkinter import simpledialog
from random import randint

class Point:
    def __init__(self, x, y):
        self.x = x
        self.y = y

class Rectangle:
    def __init__(self, x1, y1, x2, y2):
        self.x1 = min(x1, x2)
        self.y1 = min(y1, y2)
        self.x2 = max(x1, x2)
        self.y2 = max(y1, y2)

    def contains(self, point):
        return (self.x1 <= point.x <= self.x2 and
                self.y1 <= point.y <= self.y2)

class RectangleApp:
    def __init__(self, root):
        self.root = root
        self.root.title("Rectangle and Points")
        self.canvas = tk.Canvas(root, width=600, height=400, bg="white")
        self.canvas.pack()

        self.rectangle = None
        self.points = []

        self.setup_ui()

    def setup_ui(self):
        control_frame = tk.Frame(self.root)
        control_frame.pack(fill=tk.X)

        tk.Button(control_frame, text="Задать прямоугольник", command=self.set_rectangle).pack(side=tk.LEFT)
        tk.Button(control_frame, text="Добавить точки", command=self.add_points).pack(side=tk.LEFT)
        tk.Button(control_frame, text="Проверить точки", command=self.check_points).pack(side=tk.LEFT)
        tk.Button(control_frame, text="Случайные точки", command=self.random_points).pack(side=tk.LEFT)
        tk.Button(control_frame, text="Снимок", command=self.take_screenshot).pack(side=tk.LEFT)

    def set_rectangle(self):
        x1 = simpledialog.askinteger("Ввод", "Введите x1:")
        y1 = simpledialog.askinteger("Ввод", "Введите y1:")
        x2 = simpledialog.askinteger("Ввод", "Введите x2:")
        y2 = simpledialog.askinteger("Ввод", "Введите y2:")

        if all(v is not None for v in [x1, y1, x2, y2]):
            self.rectangle = Rectangle(x1, y1, x2, y2)
            self.redraw()
```

```
    def add_points(self):
        n = simpledialog.askinteger("Ввод", "Сколько точек добавить?")
        if n is not None:
            for _ in range(n):
                x = simpledialog.askinteger("Ввод", "Введите x точки:")
                y = simpledialog.askinteger("Ввод", "Введите y точки:")
                if x is not None and y is not None:
                    self.points.append(Point(x, y))
            self.redraw()

    def random_points(self):
        n = simpledialog.askinteger("Ввод", "Сколько случайных точек добавить?")
        if n is not None:
            for _ in range(n):
                x = randint(0, 600)
                y = randint(0, 400)
                self.points.append(Point(x, y))
            self.redraw()

    def check_points(self):
        if self.rectangle is None:
            return

        for point in self.points:
            if self.rectangle.contains(point):
                print(f"Точка ({point.x}, {point.y}) внутри прямоугольника")
            else:
                print(f"Точка ({point.x}, {point.y}) снаружи прямоугольника")

    def redraw(self):
        self.canvas.delete("all")

        if self.rectangle:
            self.canvas.create_rectangle(
                self.rectangle.x1, self.rectangle.y1,
                self.rectangle.x2, self.rectangle.y2,
                outline="blue", width=2
            )

        for point in self.points:
            color = "green" if (self.rectangle and self.rectangle.contains(point)) else "red"
            self.canvas.create_oval(
                point.x - 3, point.y - 3,
                point.x + 3, point.y + 3,
                fill=color, outline=color
            )

    def take_screenshot(self):
        self.canvas.postscript(file="screenshot.eps", colormode="color")
        print("Снимок сохранен как screenshot.eps")

if __name__ == "__main__":
    root = tk.Tk()
    app = RectangleApp(root)
    root.mainloop()
```

Результат выполнения:



Задание 2.

Реализовать построение заданного типа фрактала по варианту.

2) H-фрактал

Код программы:

h_fractal.py

```
import tkinter as tk
from tkinter import simpledialog

class HFractalApp:
    def __init__(self, root):
        self.root = root
        self.root.title("H-Fractal")
        self.canvas = tk.Canvas(root, width=600, height=600, bg="white")
        self.canvas.pack()

        self.depth = 5
        self.size = 200
        self.color = "black"

        self.setup_ui()
        self.draw_h_fractal()

    def setup_ui(self):
        control_frame = tk.Frame(self.root)
        control_frame.pack(fill=tk.X)

        tk.Button(control_frame, text="Изменить глубину", command=self.set_depth).pack(side=tk.LEFT)
        tk.Button(control_frame, text="Изменить размер", command=self.set_size).pack(side=tk.LEFT)
        tk.Button(control_frame, text="Изменить цвет", command=self.set_color).pack(side=tk.LEFT)
        tk.Button(control_frame, text="Снимок", command=self.take_screenshot).pack(side=tk.LEFT)

    def set_depth(self):
        depth = simpledialog.askinteger("Ввод", "Введите глубину рекурсии (1-8):", minvalue=1, maxvalue=8)
        if depth is not None:
            self.depth = depth
            self.redraw()

    def set_size(self):
        size = simpledialog.askinteger("Ввод", "Введите размер фрактала:", minvalue=50, maxvalue=300)
```

```
        if size is not None:
            self.size = size
            self.redraw()

    def set_color(self):
        color = simpledialog.askstring("Ввод", "Введите цвет (например, red, blue, #FF0000):")
        if color is not None:
            self.color = color
            self.redraw()

    def draw_h_fractal(self):
        self.canvas.delete("all")
        x, y = 300, 300
        self.draw_h(x, y, self.size, self.depth)

    def draw_h(self, x, y, size, depth):
        if depth == 0:
            return

        half = size / 2

        # Рисуем вертикальные линии
        self.canvas.create_line(x - half, y - half, x - half, y + half, fill=self.color)
        self.canvas.create_line(x + half, y - half, x + half, y + half, fill=self.color)

        # Рисуем горизонтальную линию
        self.canvas.create_line(x - half, y, x + half, y, fill=self.color)

        # Рекурсивно рисуем 4 под-фрактала
        new_size = size / 2
        new_depth = depth - 1
        self.draw_h(x - half, y - half, new_size, new_depth)
        self.draw_h(x - half, y + half, new_size, new_depth)
        self.draw_h(x + half, y - half, new_size, new_depth)
        self.draw_h(x + half, y + half, new_size, new_depth)

    def redraw(self):
        self.draw_h_fractal()

    def take_screenshot(self):
        self.canvas.postscript(file="h_fractal.eps", colormode="color")
        print("Снимок сохранен как h_fractal.eps")

if __name__ == "__main__":
    root = tk.Tk()
    app = HFractalApp(root)
    root.mainloop()
```

Результат выполнения: