Міністерство освіти і науки України

Національний університет «Запорізька політехніка»

кафедра програмних засобів

ЗВІТ

з завдання

з дисципліни «Комп'ютерна графіка та обробка зображень» на тему:

«АЛГОРИТМ КОЕНА-САЗЕРЛЕНДА»

Виконав:

ст. гр. КНТ-113сп Іван ЩЕДРОВСЬКИЙ

Прийняв:

Доцент Анжеліка ПАРХОМЕНКО

2025

1 Завдання до лабораторної роботи

Виконати програмну імітацію процедури відсікання для вікна з координатами $(l \quad r \quad b \quad t) = (30 \quad 220 \quad 50 \quad 240)$

Дано координати кінцевих точок відрізків:

Відрізок 1. P1(40, 140); P2(100, 200)

Відрізок 2. P3(10, 270); P4(300, 0)

Відрізок 3. P5(20, 10); P6(20, 200)

Відрізок 4. P7(0, 0); P8(250, 250)

2 Виконання лабораторної роботи

Для виконання роботи було створено програму-сайт, яка виконує алгоритм та візуалізує результат його виконання

В візуалізації синім показано саме вікно. За його рамками червоним кольором показано лінії, які будуть відсічені. Всередині вікна зеленим показуються лінії, які були отримані в результаті виконання алгоритму

Точки, які було отримано в результаті виконання алгоритму:

Відрізок 1. Pr1 = (40, 140), Pr2(100, 200)

Відрізок 2. Pr3 = (42.22222222222223, 240), Pr4(220, 74.48275862068967)

Відрізок 3. Pr5 = (50, 50), Pr6(220, 220)

Треба зауважити, що тут всього три відрізки, оскільки один з них повністю знаходився поза вікном

На рисунку 1 показано виконання програми

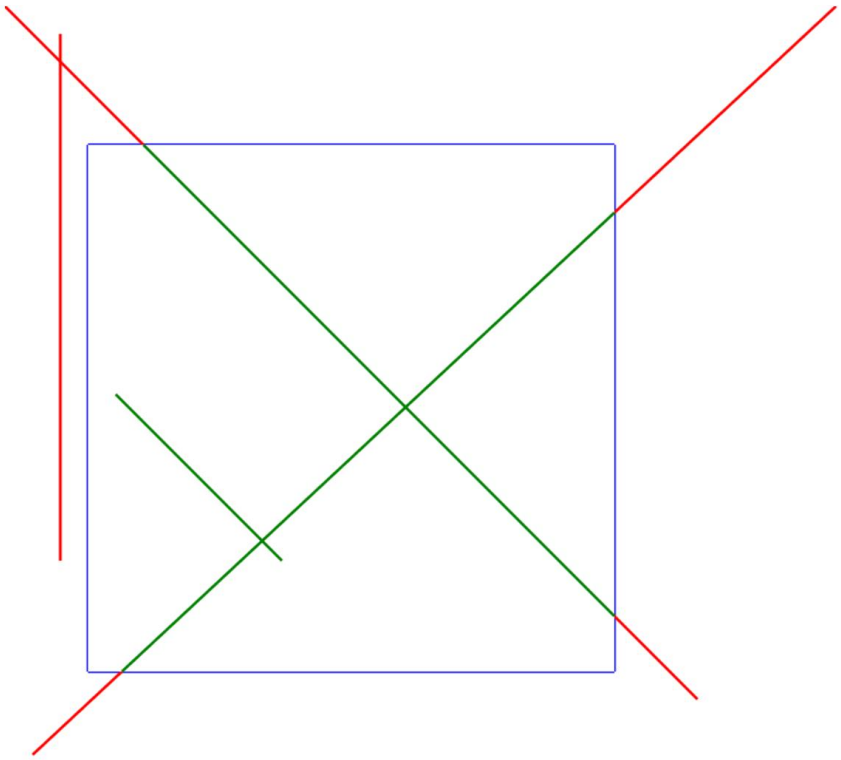

Рисунок 1 – Виконання програм

3 Тест розробленої програми

```
<!DOCTYPE html>
<html lang="en">
<head>
	<meta charset="UTF-8">
	<meta name="viewport" content="width=device-width, initial-scale=1.0">
	<title>Cohen–Sutherland</title>
</head>
<body>
	<canvas id="canvas" width="1000" height="1000"></canvas>
	<script src="main.js"></script>
</body>
</html>

const canvas = document.querySelector("#canvas")
const ctx = canvas.getContext("2d")

const RESET_INSIDE = true
const SCALE = 3

ctx.scale(SCALE, SCALE);

const display = {
	x0: 30,
	x1: 220,
	y0: 50,
	y1: 240
```

```
}

const lines = [
	{ x0: 40, y0: 140, x1: 100, y1: 200 },
	{ x0: 10, y0: 270, x1: 300, y1: 0 },
	{ x0: 20, y0: 10, x1: 20, y1: 200 },
	{ x0: 0, y0: 0, x1: 250, y1: 250 },
]

let insideLines = [
]

function drawLine(line) {
	ctx.beginPath()
	ctx.moveTo(line.x0, line.y0)
	ctx.lineTo(line.x1, line.y1)
	ctx.stroke()
	ctx.closePath()
}

function drawDisplay(display) {
	const { x0, x1, y0, y1 } = display
	ctx.strokeStyle = "blue";

	// A B
	// C D

	// C - A
	drawLine({ x0: x0, y0: y0, x1: x0, y1: y1 })

	// D - B
	drawLine({ x0: x1, y0: y0, x1: x1, y1: y1 })

	// A - B
	drawLine({ x0: x0, y0: y1, x1: x1, y1: y1 })

	// C - D
	drawLine({ x0: x0, y0: y0, x1: x1, y1: y0 })
}

function render() {
	drawDisplay(display)

	ctx.strokeStyle = "red"

	lines.map(line => {
		drawLine(line)
	})

	if (RESET_INSIDE) {
		ctx.fillStyle = "white"
		ctx.fillRect(display.x0, display.y0, display.x1 - display.x0, display.y1 - display.y0)
	}

	ctx.strokeStyle = "green"
	insideLines.map(line => {
		drawLine(line)
	})
}
```

```
const INSIDE = 0b0000;
const LEFT = 0b0001;
const RIGHT = 0b0010;
const BOTTOM = 0b0100;
const TOP = 0b1000;

function computeOutCode(display, x, y) {
	const { x0: xMin, y0: yMin, x1: xMax, y1: yMax } = display
	let code = INSIDE

	if (x < xMin) {
		code |= LEFT
	} else if (x > xMax) {
		code |= RIGHT
	}

	if (y < yMin) {
		code |= BOTTOM
	} else if (y > yMax) {
		code |= TOP
	}

	return code
}

function clipperCohenSutherland(line) {
	let { x0, y0, x1, y1 } = line
	const { x0: xMin, y0: yMin, x1: xMax, y1: yMax } = display

	let codeFirst = computeOutCode(display, x0, y0)
	let codeSecond = computeOutCode(display, x1, y1)

	while (true) {
		// Both points inside
		if ((codeFirst | codeSecond) == 0) {
			break
		}

		// Both on the same side
		if (codeFirst & codeSecond) {
			return false
		}

		let x = 0
		let y = 0

		const codeOutside = codeSecond > codeFirst ? codeSecond : codeFirst;

		// Formulas

		// d/dely = e/delx
		// dely = y1 - y0
		// delx = x1 - x0
		//
		// For right side
		// e = x1 - xMax
		// y = y1 - d
		//
```

```
                // d = e/delx * dely = (x1-xMax)/(x1-x0) * (y1-y0)
                // y = y1 - (x1-xMax) / (x1-x0) * (y1-y0)
                // y = y1 - (y1 - y0) * (x1 - xMax) / (x1 - x0)

                if (codeOutside & TOP) {
                        x = x1 - (x1 - x0) * (y1 - yMax) / (y1 - y0)
                        y = yMax;
                } else if (codeOutside & BOTTOM) {
                        x = x1 - (x1 - x0) * (y1 - yMin) / (y1 - y0)
                        y = yMin;
                } else if (codeOutside & LEFT) {
                        y = y1 - (y1 - y0) * (x1 - xMin) / (x1 - x0)
                        x = xMin
                } else if (codeOutside & RIGHT) {
                        y = y1 - (y1 - y0) * (x1 - xMax) / (x1 - x0)
                        x = xMax
                }

                if (codeOutside == codeFirst) {
                        x0 = x;
                        y0 = y;
                        codeFirst = computeOutCode(display, x0, y0);
                } else {
                        x1 = x;
                        y1 = y;
                        codeSecond = computeOutCode(display, x1, y1);
                }
        }

        return { x0, y0, x1, y1 }
}

function main() {
        lines.forEach(line => {
                const newLine = clipperCohenSutherland(line)

                if (newLine) {
                        insideLines.push(newLine)
                }
        })

        console.log(insideLines)

        render()
}

main()
```