

# ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №4

## Вариант 3 — Алгоритм средней точки и Алгоритм отсечения выпуклого многоугольника

Данный проект представляет собой приложение на **Java (Swing)**, демонстрирующее работу двух алгоритмов отсечения:

1. **Алгоритм средней точки** — используется для отсечения отрезков прямоугольным окном (вариант 3).
2. **Алгоритм отсечения выпуклого многоугольника** — применяется для отсечения выпуклого многоугольника прямоугольным окном.

Программа визуализирует исходные данные и результаты отсечения, что позволяет наглядно изучить принципы работы этих алгоритмов.

---

## Особенности реализации

- Отрисовка координатной системы с динамическим масштабированием.
- Чтение данных:
  - из файла;
  - вручную через текстовый ввод.
- Отображение:
  - исходных отрезков (синим);
  - окна отсечения (красным);
  - отсечённых частей отрезков (зелёным);
  - исходного и отсечённого многоугольника (синим и зелёным соответственно).
- Поддержка двух типов отсечения:
  - **Отрезков** — через алгоритм средней точки.
  - **Многоугольника** — через алгоритм Сазерленда–Ходжмана.

- Возможность очистки экрана и загрузки новых данных.
- 

## Формат входных данных

1. Первое число — количество отрезков  $n$ .
2. Далее для каждого отрезка:  
 $x_1 \ y_1 \ x_2 \ y_2$
3. Последняя строка — координаты прямоугольного окна:  
 $x_{min} \ y_{min} \ x_{max} \ y_{max}$ .

Пример файла:

3

10 10 200 200

50 300 400 50

150 20 150 400

0 0 300 300

---

## Используемые алгоритмы

### 1. Алгоритм средней точки

Применяется для отсечения отрезков прямоугольником (вариант 3).

Основная идея:

- Вычисляется код положения концов отрезка.
- Если отрезок полностью видим — он принимается.
- Если полностью невидим — отбрасывается.
- Если частично видим — отрезок рекурсивно делится пополам.
- Деление продолжается, пока середина не попадает на границу окна или отрезок не становится достаточно малым.

Достоинства:

- простота реализации;
- не требует сложных вычислений.

Недостатки:

- алгоритм итеративный, не самый быстрый

В проекте реализован класс `MidpointClipping`, который выполняет описанные шаги и возвращает видимую часть отрезка либо `null`.

---

## 2. Алгоритм отсечения выпуклого многоугольника

Используется для отсечения **выпуклого многоугольника**.

Алгоритм работает последовательно:

1. Многоугольник обрабатывается по одной стороне окна.
2. Для каждой стороны добавляются:
  - входящие вершины,
  - точки пересечения,
  - исключаются выходящие вершины.
3. Итог — новый многоугольник, полностью лежащий внутри окна.

В программе это реализовано в классе `SutherlandHodgmanClipping`.

---

## Интерфейс приложения

Окно приложения содержит:

- область отрисовки;
- панель кнопок:
  - **Загрузить файл**
  - **Ввести данные вручную**
  - **Отсечь отрезки (Средняя точка)**

- **Отсечь многоугольник**
  - **Очистить**
- 

## Визуализация

На холсте отображаются:

- **Система координат** — серым цветом.
- **Прямоугольное окно отсечения** — красным.  
**Исходные отрезки** — синим.
- **Отсечённые части (алгоритм средней точки)** — зелёным.
- **Исходный многоугольник** — синим.
- **Отсечённый многоугольник** — зелёным (заливка).

Масштабирование выполняется автоматически на основе минимальных и максимальных координат всех объектов.

---

## Примеры работы

### Пример 1. Частично пересекающий окно отрезок

**Ввод:**

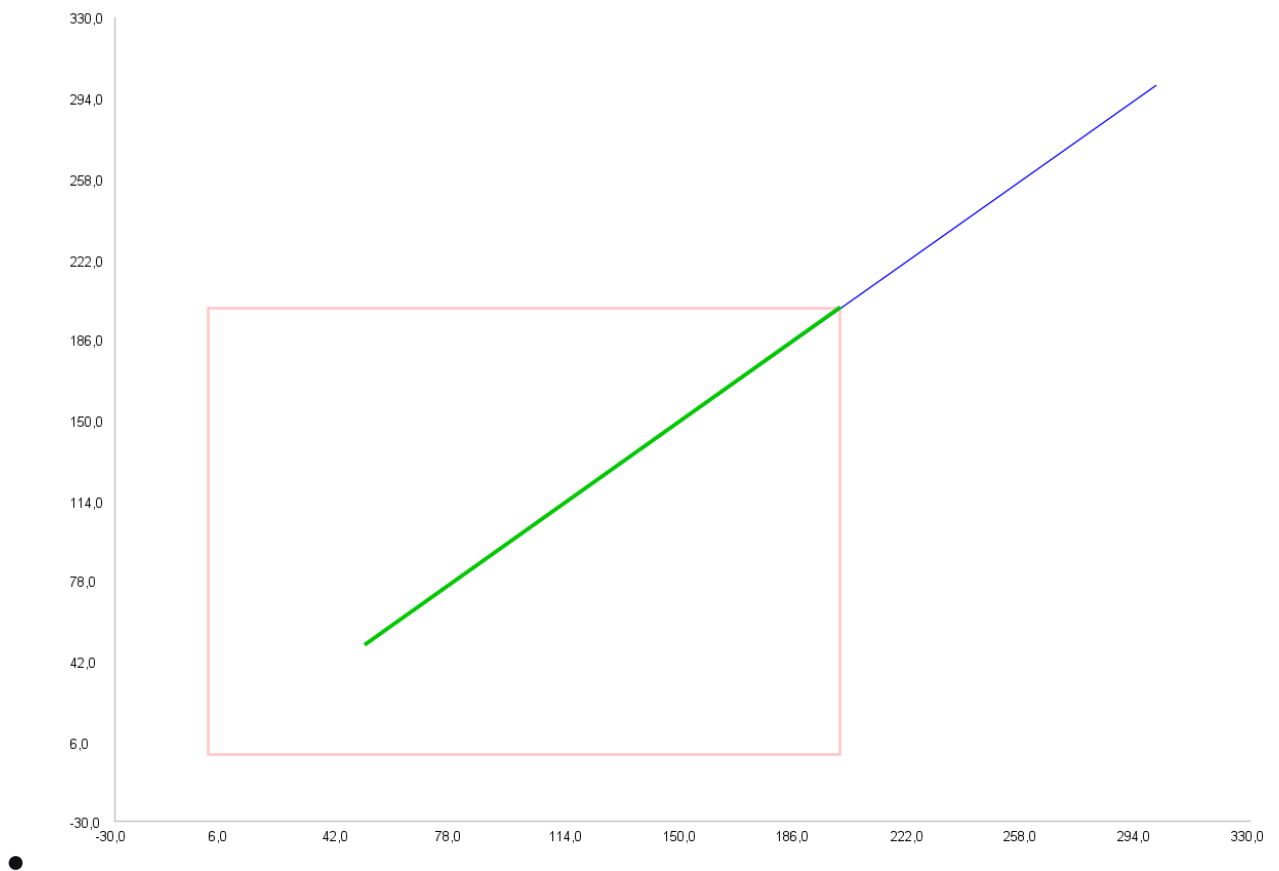
1

50 50 300 300

0 0 200 200

**Результат:**

- Синий отрезок выходит за окно.
- Алгоритм средней точки выделяет видимый участок.
- Зелёная часть показана внутри окна.



## Пример 2. Отрезок полностью внутри окна

**Ввод:**

1

50 50 150 150

0 0 200 200

**Результат:**

Отрезок полностью принимается и отображается зелёным.



### Пример 3. Отрезок полностью вне окна

**Ввод:**

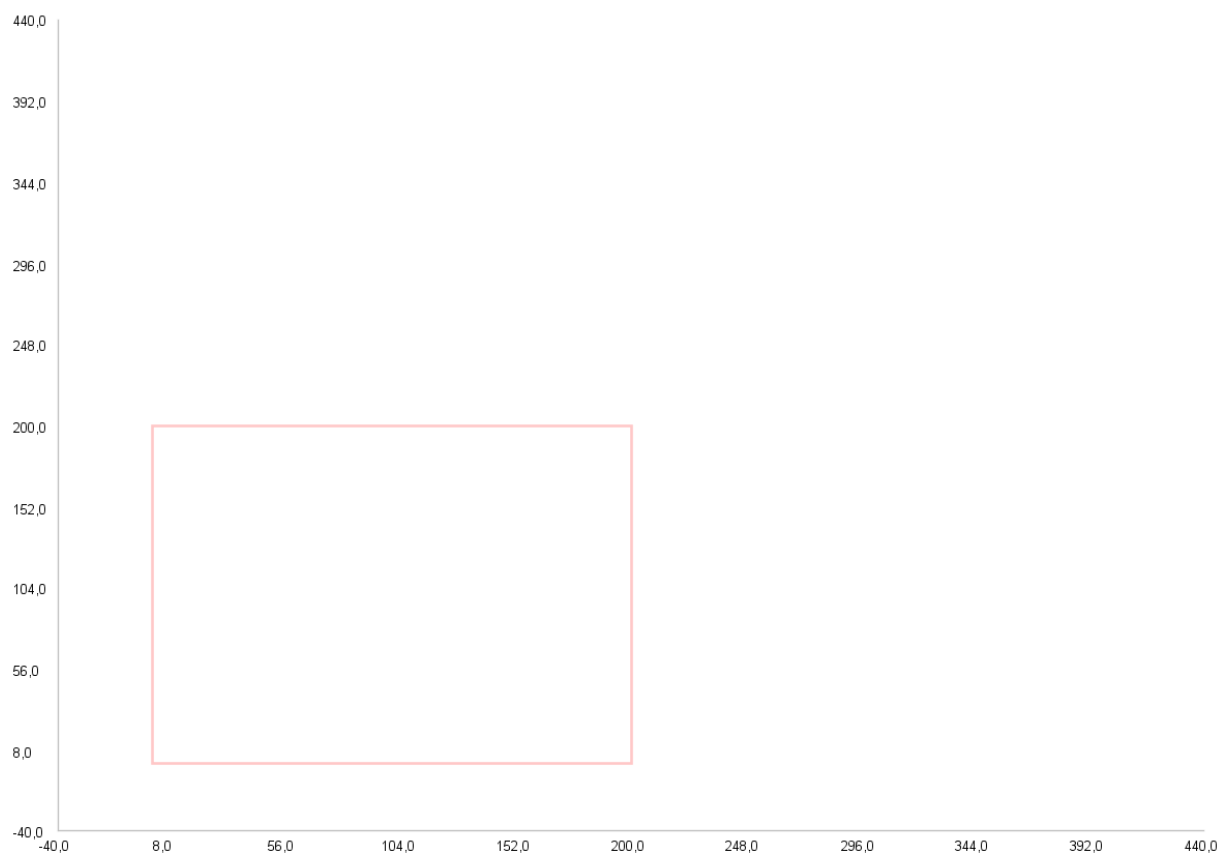
1

300 300 400 400

0 0 200 200

**Результат:**

Отрезок не отображается после отсечения.



---

#### Пример 4. Отсечение многоугольника

**Ввод:**

4

50 50 200 50

200 50 200 200

200 200 50 200

50 200 50 50

100 100 150 150

## Результат:

Алгоритм Сазерленда–Ходжмана выделяет часть многоугольника, попадающую в окно.

