ПРАКТИКУМ
Программирование символьных вычислений
«ГЕОМЕТРИЯ»

Мамиева Мария Алановна
324 группа

Москва
2025

Содержание:

0. Цель работы

Изучить базовые функциональные средства языка программирования Лисп, ориентированного на символьные вычисления и задачи искусственного интеллекта. Освоить основные приемы обработки сложных структур символьных данных, разработать и реализовать лисп-программу для определенного варианта преобразования символьных выражений.

1.Описание варианта

Геометрия
Точки на плоскости заданы с помощью Лисповского списка: ((x1 y1) (x2 y2)…).
b. По двум наборам из трех точек построить две параболы
i. определить, пересекают ли параболы ось X, если да, то найти эти точки;
ii. определить, пересекаются ли параболы, если да, найти эти точки пересечения;
iii. построить функцию, которая в любой точке X принимает значение, равное среднему арифметическому значений заданных парабол в этой точке.
c. По заданному набору точек на плоскости построить все возможные треугольники
i. найти все равнобедренные и равносторонние треугольники;
ii. найти все прямоугольные треугольники;
iii. найти все подобные треугольники.
e. По двум наборам из трех точек построить окружности
i. определить, пересекаются ли получившиеся окружности, найти эти точки пересечения;
ii. найти координаты центров обеих окружностей, определить, находится ли центр одной из них внутри другой;
iii. Если заданные окружности не пересекаются и одна не лежит внутри другой, то построить третью окружность с минимальным радиусом, которая будет касаться двух заданных и найти эти точки касания.

2. Описание кода для пункта B

Основные задачи и их реализация :

2.1 Построение парабол

Входные данные - список из трех точек, вида ((x1 y1) (x2 y2) (x3 y3)).
Вывод: Список из коэффициентов a, b, c, вида (a b c) или текст (если параболу невозможно построить).
Цель : построить парабол.
Логика:
Программа использует метод определителей (Крамера) для нахождения коэффициентов параболы вида:
$y = ax^2 + bx + c$. По трем точкам строится система уравнений:

$y_1 = a \cdot x_1^2 + b \cdot x_1 + c$
$y_2 = a \cdot x_2^2 + b \cdot x_2 + c$
$y_3 = a \cdot x_3^2 + b \cdot x_3 + c$
Коэффициенты вычисляются по формулам:
· a = det1 / full_det
· b = det2 / full_det
· c = det3 / full_det
Проверки корректности:
1. full_det ≠ 0 - точки не лежат на одной вертикали
2. det1 ≠ 0 - дополнительные проверки вырожденности
Если условия не выполняются, возвращается "Невозможно построить параболу!"

Используемые функции :

1. *build_parabola(set) - главная функция построения параболы*
2. *full_det(x1 x2 x3) - главный определитель*
3. *det1(p1 p2 p3) - определитель для коэффициента a*
4. *det2(p1 p2 p3) - определитель для коэффициента b*
5. *det3(p1 p2 p3) - определитель для коэффициента c*

Описание функций:
build_parabola(set) - подаем переменную set - лисповский список для одной параболы.
Проверяет возможность построения параболы:

- Если (full_det x1 x2 x3) = 0 → "Невозможно построить параболу!»
- Если (det1 p1 p2 p3) = 0 → "Невозможно построить параболу!»

Вычисляет коэффициенты парабол :

- a = (det1 p1 p2 p3) / (full_det x1 x2 x3)
- b = (det2 p1 p2 p3) / (full_det x1 x2 x3)
- c = (det3 p1 p2 p3) / (full_det x1 x2 x3)

full_det(x1 x2 x3) - подаем координаты x. Ищем определитель по формуле :
full_det = $(x_1 - x_2) \times (x_2 - x_3) \times (x_3 - x_1)$

det1(p1 p2 p3) - подаем три точки. Вычисляет определитель для коэффициента a по формуле :
det1 = $x_1(y_3 - y_2) - x_2(y_1 - y_3) - x_3(y_2 - y_1)$

det2(p1 p2 p3) - подаем три точки. Вычисляет определитель для коэффициента b по формуле :
det2 = $y_1(x_3^2 - x_2^2) - y_2(x_1^2 - x_3^2) - y_3(x_2^2 - x_1^2)$

det3(p1 p2 p3) - подаем три точки. Вычисляет определитель для коэффициента c по формуле :
det3 = $y_1(x_2^2 x_3 - x_3^2 x_2) - y_2(x_3^2 x_1 - x_1^2 x_3) - y_3(x_1^2 x_2 - x_2^2 x_1)$

2.2 Пересечение параболы с осью X

Входные данные: Список коэффициентов параболы вида (a b c)

Вывод:
- Если пересечений нет: "Нет пересечений параболы с осью X!"
- Если одно пересечение: (x 0) - точка пересечения
- Если два пересечения: ((x1 0) (x2 0)) - список из точек пересечения

Цель: Определить, пересекает ли парабола ось X, и найти точки пересечения.
Логика:
Парабола пересекает ось X в точках, где y = 0, т.е. решается квадратное уравнение:
$ax^2 + bx + c = 0$
Алгоритм решения:
1. Вычисляем дискриминант: $D = b^2 - 4ac$
2. Анализируем дискриминант:
   · Если D < 0: нет действительных корней - нет пересечений
   · Если D = 0: один корень - парабола касается оси X
   · Если D > 0: два корня - парабола пересекает ось X в двух точках

Формулы для вычисления корней:
$x_1 = (-b + \sqrt{D}) / (2a)$
$x_2 = (-b - \sqrt{D}) / (2a)$

Используемые функции и их описания:
x_intersect(coef) - главная функция
Вход: коэффициенты параболы (a b c)
Вызывает вспомогательную функцию с «распакованными» коэффициентами

x_intrersect_dop(a b c) - вспомогательная функция. Вычисление дискриминанта по формуле:
D = (- (* b b) (* 4 a c)) ; $D = b^2 - 4ac$
Проверка условий:
- Если D < 0: "Нет пересечений параболы с осью X!"
- Если D = 0: (один корень) x = -b/(2a), y = 0
- Если D > 0: (два корня): $x_1 = (-b + \sqrt{D})/(2a)$, $x_2 = (-b - \sqrt{D})/(2a)$

2.3 Определение пересечения двух парабол

Входные данные: Коэффициенты двух парабол вида (a1 b1 c1) и (a2 b2 c2)
Вывод:
- Совпадают: "Параболы совпадают!"
- Не пересекаются: "Параболы не пересекаются!"
- Одно пересечение: (x y)
- Два пересечения: ((x1 y1) (x2 y2))

Цель: Найти точки пересечения двух парабол.
Логика:
Точки пересечения находятся решением системы уравнений двух парабол. Приравниваем уравнения и решаем полученное квадратное уравнение относительно x, затем находим соответствующие значения y.
Алгоритм решения:
1. Вычисляем коэффициенты уравнения:
- $A = a_1 - a_2$
- $B = b_1 - b_2$

- $C = c_1 - c_2$

2. Анализируем полученное уравнение $Ax^2 + Bx + C = 0$
3. В зависимости от коэффициентов определяем количество точек пересечения

Используемые функции:

parabolas_intersect(coef1 coef2) - главная функция
· Вход: коэффициенты двух парабол
· Вычисление: коэффициенты A, B, C разностного уравнения
· Вызов: функции поиска пересечений с вычисленными коэффициентами

intersect_search(coef A B C) - поиск пересечений
Анализ особых случаев:
- A=0, B=0, C=0: "Параболы совпадают!»
- A=0, B=0:"Параболы не пересекаются!"
- A=0: линейное уравнение, одно решение, x = -C/B и y (вычисляем при помощи функции calculate_y coef).
- A≠0: квадратное уравнение, два решения, вызываем функцию discrim_check coef

discrim_check(coef A B C) - проверка дискриминанта
Вычисление дискриминанта: $D = b^2 - 4ac$
Анализ дискриминанта:
- D < 0: нет действительных корней
- D = 0: один корень (касание)
- D > 0: два корня (пересечение)

Вычисление координат точек пересечения

calculate_y(coef x) - вычисление y-координаты
Вычисляет значение y для заданного x по формуле параболы: $y = ax^2 + bx + c$

2.4 Построение функции среднего арифметического парабол

Входные данные: Коэффициенты двух парабол вида (a1 b1 c1) и (a2 b2 c2)
Вывод: Уравнение функции в виде списка ("Y=" a_avg "x^2+" b_avg "x+" c_avg) с упрощением при нулевых коэффициентах
Цель: Построить функцию, которая в любой точке X принимает значение, равное среднему арифметическому значений заданных парабол в этой точке.
Логика:
Среднее арифметическое двух парабол $y_1(x) = a_1x^2 + b_1x + c_1$ и $y_2(x) = a_2x^2 + b_2x + c_2$ вычисляется по формуле:
$Y\_avg(x) = (y_1(x) + y_2(x)) / 2 = ((a_1 + a_2)/2)x^2 + ((b_1 + b_2)/2)x + ((c_1 + c_2)/2)$
Алгоритм:
1. Вычисляем средние коэффициенты:
   · $a\_avg = (a_1 + a_2) / 2$
   · $b\_avg = (b_1 + b_2) / 2$
   · $c\_avg = (c_1 + c_2) / 2$
2. Форматируем уравнение с учетом нулевых коэффициентов

Используемая функция:

arith_mean(coef1 coef2) - построение средней функции

Проверка особых случаев упрощения:
- Если a_avg = 0 и b_avg = 0: функция постоянна "Y=" c_avg
- Если a_avg = 0: функция линейна "Y=" b_avg "x+" c_avg
- Общий случай: полное квадратное уравнение "Y=" a_avg "x^2+" b_avg "x+" c_avg)

2.5 Координация работы пункта B

2.5.1 Главная функция координации

parB(l1 l2) - главная функция пункта B
Входные данные: Два списка точек для двух парабол вида ((x1 y1) (x2 y2) (x3 y3))
Вывод: Полный анализ обеих парабол с промежуточными результатами
Логика: Координирует весь процесс анализа двух парабол, последовательно вызывая функции построения и анализа.
Пошаговое описание:
I. Вывод заголовка "Пункт B"
II. Построение и вывод первой параболы («Парабола 1», (build_parabola l1))
III. Построение и вывод второй параболы («Парабола 2», (build_parabola l2))
IV. Передача результатов построения функции анализа(bdop (build_parabola l1) (build_parabola l2))

2.5.2 Функция анализа результатов

bdop(coef1 coef2) - обработчик результатов построения парабол
Входные данные: Коэффициенты двух построенных парабол
Вывод: Полный анализ пересечений и построение средней функции
Логика: Анализирует результаты построения парабол и в зависимости от их корректности выполняет соответствующий анализ.
Пошаговое описание:
Обе параболы построены успешно:
Анализ пересечений с осью X для обеих парабол
Анализ пересечений между параболами
Построение средней функции
Только первая парабола построена успешно: анализ только первой параболы (пересечения с осью X)
Только вторая парабола построена успешно: анализ только второй параболы (пересечения с осью X)

3. Описание кода для пункта C

3.1 Построение всех возможных треугольников

Входные данные: Список точек на плоскости вида ((x1 y1) (x2 y2) (x3 y3) ...)
Вывод: Список всех возможных невырожденных треугольников вида (((x1 y1) (x2 y2) (x3 y3)) …)
Цель: Построить все возможные треугольники из заданного набора точек, исключая вырожденные случаи.
Логика: Программа генерирует все возможные комбинации по 3 точки из заданного набора, затем фильтрует их, оставляя только невырожденные треугольники (с ненулевой площадью).
Алгоритм:
- Удаление дублирующихся точек
- Генерация всех комбинаций по 3 точки
- Проверка каждой комбинации на невырожденность (площадь ≠ 0)
- Возврат списка валидных треугольников

Используемые функции:
build_triangle(l) - главная функция построения треугольников
Вход: список точек
Вызов: удаление дубликатов, генерация комбинаций, проверка треугольников
Вывод: список всех возможных треугольников

rem_dup(l) - удаление дублирующихся точек.
Рекурсивно удаляет повторяющиеся точки из списка.
Использует: find_dup для проверки наличия дубликатов

find_dup(e l) - поиск дубликатов в списке.
Проверяет наличие элемента в списке.
Возвращает: T если найден, nil если нет (аналог member)

build_triangles(l) - построение всех комбинаций треугольников.
Обрабатывает список комбинаций точек.
Для каждой комбинации вызывает проверку валидности.
Объединяет результаты рекурсивно

check_triangles(tri) - проверка треугольника.
Проверяет что треугольник не вырожденный.
Использует: area для вычисления площади. Возвращает: треугольник если площадь ≠ 0, иначе nil.

area(tri) - вычисление площади треугольника.
Использует формулу площади через координаты вершин:
$$area = |(x_1(y_2 - y_3) + x_2(y_3 - y_1) + x_3(y_1 - y_2)) / 2|$$
Если площадь = 0 - треугольник вырожден

free_points(l) - генерация всех комбинаций точек.
Генерирует все комбинации по 3 точки рекурсивно.
Использует: free_dop для фиксации первой точки

free_dop(a l) - вспомогательная для комбинаций.
Фиксирует первую точку a.
Генерирует комбинации с оставшимися точками

free_ddop(a b l) - вспомогательная для комбинаций.
Фиксирует первые две точки a и b.
Добавляет третью точку из оставшегося списка.
Создает треугольник (a b c) для каждой точки c

3.2 Поиск равнобедренных и равносторонних треугольников

Входные данные: Список треугольников вида (((x1 y1) (x2 y2) (x3 y3)) ...)
Вывод:

- Равнобедренные треугольники: список треугольников с двумя равными сторонами
- Равносторонние треугольники: список треугольников со всеми равными сторонами

Цель: Найти все равнобедренные и равносторонние треугольники среди построенных.
Логика: Программа проверяет каждый треугольник на равенство длин сторон с использованием приблизительного сравнения для учета погрешностей вычислений.
Используемые функции:
distance(p1 p2) - вычисление расстояния между точками. Использует формулу расстояния: $\sqrt{((x_2 - x_1)^2 + (y_2 - y_1)^2)}$
Вычисляет длину стороны треугольника

approx_equal(a b) - проверка приблизительного равенства.
Проверяет равенство чисел с погрешностью 0.001.
Учитывает погрешности вычислений с плавающей точкой.

equilateral(l) - поиск равносторонних треугольников.
Фильтрует список , оставляя только равносторонние.
Использует: equilateral1 для проверки каждого треугольника

equilateral1(l) - проверка равносторонности.
Проверяет что все три стороны равны.
Возвращает: T если треугольник равносторонний

isosceles(l) - поиск равнобедренных треугольников.
Фильтрует список треугольников, оставляя только равнобедренные.
Использует: isosceles1 для проверки каждого треугольника

isosceles1(l) - проверка равнобедренности.
Проверяет что хотя бы две стороны равны.
Возвращает: T если треугольник равнобедренный

3.3 Поиск прямоугольных треугольников

Входные данные: Список треугольников вида (((x1 y1) (x2 y2) (x3 y3)) ...)
Вывод: Список прямоугольных треугольников (треугольников с прямым углом)
Цель: Найти все прямоугольные треугольники среди построенных.

Логика: Программа проверяет каждый треугольник на выполнение теоремы Пифагора - что квадрат одной стороны равен сумме квадратов двух других сторон.
Пошаговый алгоритм проверки:

I. Вычисление длин сторон треугольника
II. Проверка трех вариантов теоремы Пифагора
III. Если выполняется хотя бы одно условие - треугольник прямоугольный

Используемые функции:

right_triangles(l) - поиск прямоугольных треугольников
Фильтрует список треугольников
Оставляет только прямоугольные
Использует: right_triangle_p для проверки каждого треугольника

right_triangle_p(tri) - проверка прямоугольности треугольника
Вычисляет длины всех трех сторон
Передает стороны функции right_triangle_p1 для проверки теоремы Пифагора

right_triangle_p1(a b c) - проверка теоремы Пифагора
Проверяет все три возможные гипотенузы:
- Если $a^2 + b^2 \approx c^2$ (c - гипотенуза)
- Если $a^2 + c^2 \approx b^2$ (b - гипотенуза)
- Если $b^2 + c^2 \approx a^2$ (a - гипотенуза)

Использует: approx_equal для учета погрешностей вычислений

3.4 Поиск подобных треугольников

Входные данные: Список треугольников вида (((x1 y1) (x2 y2) (x3 y3)) ...)
Вывод: Список пар подобных треугольников вида ((t1 t2) (t3 t4) ...)
Цель: Найти все пары подобных треугольников среди построенных.
Логика: Программа проверяет все возможные пары треугольников на подобие, сравнивая отношения длин их сторон после сортировки.
Пошаговый алгоритм проверки подобия:
I. Для каждого треугольника t1:
- Вычислить длины всех трех сторон
- Отсортировать стороны по возрастанию: (s1 s2 s3)

II. Для каждого треугольника t2 (из оставшихся):Вычислить и отсортировать его стороны: (s1' s2' s3')
III. Проверить пропорциональность
IV. Если пропорциональность выполняется - треугольники подобны
Используемые функции:

similar_triangles(l) - поиск всех подобных треугольников
Рекурсивно обрабатывает список треугольников
Для каждого треугольника ищет подобные среди оставшихся
Объединяет результаты с рекурсивным вызовом для хвоста списка

find_similar(t1 l) - поиск подобных для одного треугольника
Ищет все треугольники в списке l, подобные треугольнику t1
Возвращает список пар (t1 t2) для каждой найденной пары

similar_p(t1 t2) - проверка подобия двух треугольников
Вычисляет и сортирует стороны обоих треугольников
Сравнивает отношения сторон через функцию similar_check

similar_check(s1 s2) - проверка пропорциональности сторон
Проверяет что все отношения сторон равны:
Защита от деления на ноль: проверяет что $s2_1 > 0$

sort3(a b c) - сортировка трех чисел
Сортирует три числа по возрастанию
Обеспечивает корректное сравнение сторон треугольников

3.5 Координация работы пункта C

Главная функция координации - triangleC(l)
Входные данные: Список точек на плоскости вида ((x1 y1) (x2 y2) (x3 y3) ...)
Вывод: Полный анализ всех треугольников с классификацией по типам
Цель: Координирует весь процесс построения и анализа треугольников, последовательно вызывая функции построения и классификации.
Логика: Функция последовательно строит все треугольники, затем находит и выводит все типы треугольников: обычные, равнобедренные, равносторонние, прямоугольные и подобные.
Пошаговое описание:

I. Вывод заголовка пункта
II. Построение и вывод всех треугольников
III. Поиск и вывод равнобедренных треугольников
IV. Поиск и вывод равносторонних треугольников
V. Поиск и вывод прямоугольных треугольников
VI. Поиск и вывод подобных треугольников

4. Описание кода для пункта E

4.1 Построение окружности по трем точкам

Входные данные: Список из трех точек вида ((x1 y1) (x2 y2) (x3 y3))
Вывод:

- При успехе: ((center_x center_y) radius) - центр и радиус окружности

- При ошибке: "Невозможно построить окружность!"

Цель: Построить окружность, проходящую через три заданные точки.
Логика: Программа использует геометрический метод для нахождения центра окружности через пересечение серединных перпендикуляров, затем вычисляет радиус как расстояние от центра до любой из точек.
Алгоритм:

I. Проверка что точки не коллинеарны
II. Вычисление координат центра окружности
III. Вычисление радиуса как расстояния от центра до любой точки

Используемые функции:
circle_build(l) - главная функция построения окружности
Пошаговое описание circle_build:
Проверка возможности построения - ((not (can_build (car l) (cadr l) (caddr l))) "Невозможно построить окружность!")Использует: can_build для проверки что точки не коллинеарны
Вычисление координат центра, используя formula_for_circle
Вычисление радиуса, используя radius

formula_for_circle(a b c d e f) - вычисление координаты центра
Для точек A(x1,y1), B(x2,y2), C(x3,y3) координата X центра:
$X = [(y2-y1)(y3^2-y1^2 + x3^2-x1^2) - (y3-y1)(y2^2-y1^2 + x2^2-x1^2)] / [2((x3-x1)(y2-y1) - (x2-x1)(y3-y1))]$

radius(x y point) - вычисление радиуса
Формула радиуса: $\sqrt{((x\_center - x\_point)^2 + (y\_center - y\_point)^2)}$

can_build(p1 p2 p3) - проверка возможности построения
Проверяет что точки не коллинеарны через векторное произведение
Если результат = 0 - точки на одной прямой, нельзя построить окружность

4.2 Определение пересечения двух окружностей

Входные данные: Две окружности вида ((center_x1 center_y1) radius1) и ((center_x2 center_y2) radius2)
Вывод:

- Совпадают: "Окружности совпадают"
- Не пересекаются: "Окружности не пересекаются"
- Одна внутри другой: "Одна окружность внутри другой"
- Касаются: ("Окружности касаются" (x y))
- Пересекаются: ("Окружности пересекаются в двух точках" ((x1 y1) (x2 y2))

Цель: Определить взаимное расположение двух окружностей и найти точки пересечения если они существуют.
Логика:Программа анализирует расстояние между центрами окружностей и их радиусы, чтобы определить тип пересечения, затем вычисляет координаты точек пересечения.
Математическая основа: точки пересечения находятся через параметры a и h:

$P_1 = (x1 + a \cdot dx/d \pm h \cdot dy/d, y1 + a \cdot dy/d \mp h \cdot dx/d)$

$P_2 = (x1 + a \cdot dx/d \mp h \cdot dy/d, y1 + a \cdot dy/d \pm h \cdot dx/d)$

Используемые функции:
circles_relation(circle1 circle2) - главная функция определения отношений
Пошаговое описание circles_relation:

I. Проверка совпадения окружностей: центры совпадают И радиусы равны
II. Проверка концентрических окружностей: центры совпадают, но радиусы разные
III. Проверка непересекающихся окружностей: расстояние между центрами > суммы радиусов
IV. Проверка вложенных окружностей: расстояние между центрами < |r1 - r2|
V. Проверка касания: расстояние = сумме радиусов ИЛИ расстояние = |r1 - r2|
VI. Пересечение в двух точках, все остальные случаи: использует find_points

find_points(x1 y1 r1 x2 y2 r2) - нахождение точек пересечения

Назначение: Вычисление координат точек пересечения двух окружностей
Входные данные: x1, y1, r1 - координаты центра и радиус первой окружности,x2, y2, r2 - координаты центра и радиус второй окружности
Вывод: Список из двух точек пересечения ((x1 y1) (x2 y2))
Функция использует **геометрический метод** нахождения точек пересечения через параметры a и h.

calc_a(x1 y1 r1 x2 y2 r2) - вычисление параметра a - расстояние до хорды
Формула: $a = (r_1^2 - r_2^2 + d^2) / (2d)$, где d - расстояние между центрами

calc_h(x1 y1 r1 x2 y2 r2) - вычисление параметра h - половина хорды
Формула: $h = \sqrt{(r_1^2 - a^2)}$

4.3 Анализ центров окружностей и их взаимного расположения

Входные данные: Две окружности вида ((center_x1 center_y1) radius1) и ((center_x2 center_y2) radius2)
Вывод: Информация о центрах и их взаимном расположении в виде списка
Цель: Найти координаты центров обеих окружностей и определить, находится ли центр одной из них внутри другой.
Используемые функции:
analyze(c1 c2) - главная функция анализа
Пошаговое описание analyze:

I. Вывод координат центра первой окружности
II. Вывод координат центра второй окружности
III. Определение взаимного расположения центров, использует inside

inside(c1 c2) - проверка вложенности центров
Пошаговое описание inside:

I. Проверка что ни одна окружность не находится внутри другой. Условие: расстояние между центрами ≥ |r1 - r2|. Геометрический смысл: центры достаточно далеко друг от друга, чтобы одна окружность не могла полностью содержать другую
II. Проверка что первая окружность внутри второй. Условие: r1 < r2 И расстояние < (r2 - r1).Геометрический смысл: меньшая окружность полностью содержится в большей

III. Вторая окружность внутри первой.Условие: r2 < r1 И расстояние < (r1 - r2).
Геометрический смысл: вторая окружность полностью содержится в первой

Математическая основа:Критерий вложенности окружностей:

Окружность A находится внутри окружности B, если:

distance(center_A, center_B) < |radius_B - radius_A|

Учет особых случаев:

1. Когда центры совпадают - сравниваются радиусы
2. Когда расстояние между центрами ≤ разности радиусов - одна окружность полностью внутри другой
3. Все остальные случаи - окружности пересекаются или разделены

4.4 Построение касательной окружности

Входные данные: Два списка по три точки каждый, вида ((x1 y1) (x2 y2) (x3 y3))

Вывод:

При успехе:

Центр и радиус касательной окружности в виде ((x y) r)

Точки касания с исходными окружностями

При ошибке: "Нельзя построить касательную окружность"

Цель: Построить третью окружность с минимальным радиусом, которая касается двух заданных окружностей.

Логика: Программа строит две исходные окружности, проверяет их взаимное расположение, и если окружности не пересекаются и не вложены друг в друга, строит касательную окружность с минимальным радиусом.

Алгоритм:

I. Построение двух окружностей по трем точкам каждая
II. Проверка что окружности не пересекаются и не вложены
III. Вычисление центра и радиуса касательной окружности
IV. Нахождение точек касания с исходными окружностями

Используемые функции:

tangent_check(circle1 circle2) - главная функция проверки и построения

Пошаговое описание tangent_check:

I. Проверка взаимного расположения.Использует: circles_relation для проверки пересечения, inside для проверки вложенности
II. Если условия выполняются → вызывает tangent_circle для построения
III. Иначе возвращает "Нельзя построить касательную окружность"

circles_relation(circle1 circle2) - анализ взаимного расположения:

Проверяет расстояние между центрами и радиусы

Возвращает: "Окружности совпадают", "Одна внутри другой", "Не пересекаются", "Касаются" или "Пересекаются"

inside(circle1 circle2) - проверка вложенности:

Сравнивает расстояние между центрами с разностью радиусов

Возвращает какая окружность внутри или "Ни одна не внутри"

tangent_circle(circle1 circle2) - построение касательной окружности:

Вычисляет центр через tangent_center

Вычисляет радиус через tangent_radius
Находит точки касания через find_points

tangent_center(circle1 circle2) - вычисление центра:
Находит точку на линии между центрами исходных окружностей

tangent_radius(circle1 circle2) - вычисление радиуса:
Формула: r = (d - r1 - r2) / 2, где d - расстояние между центрами

find_points(x1 y1 r1 x2 y2 r2) - нахождение точек касания:
Использует геометрические соотношения для вычисления координат точек касания

4.5 Координация пункта E

Входные данные: Два списка по три точки каждый, вида ((x1 y1) (x2 y2) (x3 y3))
Вывод:
Полный анализ двух окружностей и их касательной окружности
Поэтапные результаты построения и вычислений
Цель: Координировать весь процесс анализа двух окружностей и построения касательной окружности между ними.
Логика: Программа последовательно выполняет построение двух окружностей, анализирует их взаимное расположение, проверяет условия для построения касательной окружности и выводит все промежуточные результаты.
Алгоритм:
I. Построение первой окружности по трем точкам
II. Построение второй окружности по трем точкам
III. Анализ взаимного расположения окружностей
IV. Проверка условий для построения касательной окружности
V. Построение касательной окружности (если условия выполняются)
VI. Вывод всех результатов
Используемые функции:
punE(l1 l2) - главная функция координации пункта E
Пошаговое описание punE:
I. Вывод заголовка "Пункт E"
II. Построение и вывод первой окружности: (print "Окружность 1") (print (circle_build l1))
III. Построение и вывод второй окружности: (print "Окружность 2") (print (circle_build l2))
IV. Вызов функции анализа и построения: (edop (circle_build l1) (circle_build l2))

edop(circle1 circle2) - вспомогательная функция анализа и построения
Пошаговое описание edop:
I. Проверка успешности построения обеих окружностей: ((and (listp circle1) (listp circle2)) ...)
II. Анализ отношения окружностей: (print "Отношение окружностей") (print (circles_relation circle1 circle2))
III. Анализ центров: (print "Анализ центров") (print (analyze circle1 circle2))
IV. Построение касательной окружности: (print "Касательная окружность") (tangent_check circle1 circle2)
V. Обработка случаев когда построена только одна окружность

VI. Обработка ошибок построения

5.Итог пункт B

5.1 Тесты для пункта B

```
;; Тест 1: Нормальное построение двух парабол - проверка основного функционала
(print "=== Тест 1: Нормальное построение двух парабол ===")
(print "Цель: Проверить корректное построение парабол и их анализ")
(parB '((0 0) (1 1) (2 4)) '((0 1) (1 0) (2 1)))

;; Тест 2: Параболы не пересекаются - проверка обработки непересекающихся кривых
(print "=== Тест 2: Параболы не пересекаются ===")
(print "Цель: Проверить обработку случая непересекающихся парабол")
(parB '((0 2) (1 3) (2 6)) '((0 -1) (1 -2) (2 -5)))

;; Тест 3: Параболы касаются - проверка граничного случая касания
(print "=== Тест 3: Параболы касаются ===")
(print "Цель: Проверить случай близких, но не пересекающихся парабол")
(parB '((0 0) (1 1) (2 4)) '((0 1) (1 2) (2 5)))

;; Тест 4: Вырожденный случай - точки на одной прямой - проверка обработки ошибок
(print "=== Тест 4: Вырожденный случай - точки на одной прямой ===")
(print "Цель: Проверить обработку невозможности построения параболы")
(parB '((0 0) (1 1) (2 2)) '((0 1) (1 2) (2 3)))

;; Тест 5: Параболы совпадают - проверка идентичных кривых
(print "=== Тест 5: Параболы совпадают ===")
(print "Цель: Проверить обработку идентичных парабол")
(parB '((0 0) (1 1) (2 4)) '((0 0) (1 1) (2 4)))

;; Тест 6: Одна точка пересечения - проверка вырожденного пересечения
(print "=== Тест 6: Одна точка пересечения ===")
(print "Цель: Проверить нахождение одной точки пересечения")
(parB '((0 0) (1 1) (2 4)) '((0 0) (1 0) (2 0)))

;; Тест 7: Вертикальные "параболы" - проверка особых случаев построения
(print "=== Тест 7: Вертикальные параболы ===")
(print "Цель: Проверить обработку вертикальных линий")
(parB '((0 0) (0 1) (0 4)) '((1 0) (1 1) (1 4)))

;; Тест 8: Комплексные корни - проверка парабол без действительных корней
(print "=== Тест 8: Комплексные корни ===")
(print "Цель: Проверить обработку парабол без действительных пересечений с осью X")
(parB '((0 2) (1 3) (2 6)) '((0 1) (1 2) (2 5)))

;; Тест 9: Большие числа - проверка устойчивости к большим значениям
```

```
(print "=== Тест 9: Большие числа ===")
(print "Цель: Проверить устойчивость к большим значениям")
(parB '((100 10000) (200 40000) (300 90000)) '((100 5000) (200 10000) (300 15000)))

;; Тест 10: Отрицательные координаты - проверка работы с отрицательными значениями
(print "=== Тест 10: Отрицательные координаты ===")
(print "Цель: Проверить работу с отрицательными координатами")
(parB '((-2 4) (-1 1) (0 0)) '((-2 1) (-1 0) (0 1)))

;; Тест 11: Только одна парабола строится - проверка частичного успеха
(print "=== Тест 11: Только одна парабола строится ===")
(print "Цель: Проверить обработку частично успешного построения")
(parB '((0 0) (1 1) (2 2)) '((0 0) (1 1) (2 4)))

;; Тест 12: Горизонтальные линии - проверка вырожденных парабол
(print "=== Тест 12: Горизонтальные линии ===")
(print "Цель: Проверить построение горизонтальных линий")
(parB '((0 0) (1 0) (2 0)) '((0 1) (1 1) (2 1)))

;; Тест 13: Симметричные параболы - проверка симметричных случаев
(print "=== Тест 13: Симметричные параболы ===")
(print "Цель: Проверить обработку симметричных парабол")
(parB '((-1 1) (0 0) (1 1)) '((-1 2) (0 1) (1 2)))

;; Тест 14: Параболы с разными направлениями - проверка разнонаправленных кривых
(print "=== Тест 14: Параболы с разными направлениями ===")
(print "Цель: Проверить параболы с разной выпуклостью")
(parB '((0 0) (1 1) (2 4)) '((0 4) (1 1) (2 0)))
```

5.2 Ответы на тесты

```
"=== Тест 1: Нормальное построение двух парабол ==="
"Цель: Проверить корректное построение парабол и их анализ"
"Пункт в"
"Парабола 1"
(1 0 0)
"Парабола 2"
(1 1 -1)
"Пересечение с осью x параболы 1"
((0 0))
"Пересечение с осью x параболы 2"
((0.618034 0) (-1.618034 0))
"Пересечение парбол (точки если есть)"
((1 1))
"Функция, которая в любой точке X принимает значение, равное
среднему арифметическому значений заданных парабол в этой точке"
("Y=" 1 "x^2+" 1/2 "x+" -1/2)
"=== Тест 2: Параболы не пересекаются ==="
"Цель: Проверить обработку случая непересекающихся парабол"
```

```
"Пункт B"
"Парабола 1"
(1 6 -2)
"Парабола 2"
(-1 -3 1)
"Пересечение с осью x параболы 1"
((0.31662488 0) (-6.3166246 0))
"Пересечение с осью x параболы 2"
((-3.3027756 0) (0.30277562 0))
"Пересечение парбол (точки если есть)"
((0.31173778 -0.03239286) (-4.811738 -7.7176056))
"Функция, которая в любой точке X принимает значение, равное
среднему арифметическому значений заданных парабол в этой точке"
("Y=" 3/2 "x+" -1/2)
"=== Тест 3: Параболы касаются ==="
"Цель: Проверить случай близких, но не пересекающихся парабол"
"Пункт B"
"Парабола 1"
(1 0 0)
"Парабола 2"
(1 3 -1)
"Пересечение с осью x параболы 1"
((0 0))
"Пересечение с осью x параболы 2"
((0.30277562 0) (-3.3027756 0))
"Пересечение парбол (точки если есть)"
((1/3 1/9))
"Функция, которая в любой точке X принимает значение, равное
среднему арифметическому значений заданных парабол в этой точке"
("Y=" 1 "x^2+" 3/2 "x+" -1/2)
"=== Тест 4: Вырожденный случай - точки на одной прямой ==="
"Цель: Проверить обработку невозможности построения параболы"
"Пункт B"
"Парабола 1"
"Невозможно построить параболу!"
"Парабола 2"
"Невозможно построить параболу!"
"=== Тест 5: Параболы совпадают ==="
"Цель: Проверить обработку идентичных парабол"
"Пункт B"
"Парабола 1"
(1 0 0)
"Парабола 2"
(1 0 0)
"Пересечение с осью x параболы 1"
((0 0))
"Пересечение с осью x параболы 2"
((0 0))
"Пересечение парбол (точки если есть)"
"Параболы совпадают!"
```

```
"Функция, которая в любой точке X принимает значение, равное
среднему арифметическому значений заданных парабол в этой точке"
("Y=" 1 "x^2+" 0 "x+" 0)
"=== Тест 6: Одна точка пересечения ==="
"Цель: Проверить нахождение одной точки пересечения"
"Пункт B"
"Парабола 1"
(1 0 0)
"Парабола 2"
"Невозможно построить параболу!"
"Пересечение с осью x параболы 1"
((0 0))
"=== Тест 7: Вертикальные параболы ==="
"Цель: Проверить обработку вертикальных линий"
"Пункт B"
"Парабола 1"
"Невозможно построить параболу!"
"Парабола 2"
"Невозможно построить параболу!"
"=== Тест 8: Комплексные корни ==="
"Цель: Проверить обработку парабол без действительных пересечений
с осью X"
"Пункт B"
"Парабола 1"
(1 6 -2)
"Парабола 2"
(1 3 -1)
"Пересечение с осью x параболы 1"
((0.31662488 0) (-6.3166246 0))
"Пересечение с осью x параболы 2"
((0.30277562 0) (-3.3027756 0))
"Пересечение парбол (точки если есть)"
((1/3 1/9))
"Функция, которая в любой точке X принимает значение, равное
среднему арифметическому значений заданных парабол в этой точке"
("Y=" 1 "x^2+" 9/2 "x+" -3/2)
"=== Тест 9: Большие числа ==="
"Цель: Проверить устойчивость к большим значениям"
"Пункт B"
"Парабола 1"
(6 500 -60000)
"Парабола 2"
(1/2 300 -30000)
"Пересечение с осью x параболы 1"
((200/3 0) (-150 0))
"Пересечение с осью x параболы 2"
((87.29834 0) (-687.29834 0))
"Пересечение парбол (точки если есть)"
((57.878185 -10961.602) (-94.24182 -53831.785))
"Функция, которая в любой точке X принимает значение, равное
среднему арифметическому значений заданных парабол в этой точке"
```

```
("Y=" 13/4 "x^2+" 400 "x+" -45000)
"=== Тест 10: Отрицательные координаты ==="
"Цель: Проверить работу с отрицательными координатами"
"Пункт B"
"Парабола 1"
(3 -4 0)
"Парабола 2"
(-1 1 1)
"Пересечение с осью x параболы 1"
((4/3 0) (0 0))
"Пересечение с осью x параболы 2"
((-0.618034 0) (1.618034 0))
"Пересечение парбол (точки если есть)"
((1.4253905 0.39365244) (-0.17539054 0.7938477))
"Функция, которая в любой точке X принимает значение, равное
среднему арифметическому значений заданных парабол в этой точке"
("Y=" 1 "x^2+" -3/2 "x+" 1/2)
"=== Тест 11: Только одна парабола строится ==="
"Цель: Проверить обработку частично успешного построения"
"Пункт B"
"Парабола 1"
"Невозможно построить параболу!"
"Парабола 2"
(1 0 0)
"Пересечение с осью x параболы 2"
((0 0))
"=== Тест 12: Горизонтальные линии ==="
"Цель: Проверить построение горизонтальных линий"
"Пункт B"
"Парабола 1"
"Невозможно построить параболу!"
"Парабола 2"
"Невозможно построить параболу!"
"=== Тест 13: Симметричные параболы ==="
"Цель: Проверить обработку симметричных парабол"
"Пункт B"
"Парабола 1"
"Невозможно построить параболу!"
"Парабола 2"
"Невозможно построить параболу!"
"=== Тест 14: Параболы с разными направлениями ==="
"Цель: Проверить параболы с разной выпуклостью"
"Пункт B"
"Парабола 1"
(1 0 0)
"Парабола 2"
(1 8 -4)
"Пересечение с осью x параболы 1"
((0 0))
"Пересечение с осью x параболы 2"
((0.47213602 0) (-8.472136 0))
```

```
"Пересечение парабол (точки если есть)"
((1/2 1/4))
"Функция, которая в любой точке X принимает значение, равное
среднему арифметическому значений заданных парабол в этой точке"
("Y=" 1 "x^2+" 4 "x+" -2)
```

Выводы: Все тесты работают верно

5.3 Текст кода пункта B

```
(defun parB (l1 l2)
        (print "Пункт B")
        (print "Парабола 1")
        (print (build_parabola l1))
        (print "Парабола 2")
        (print (build_parabola l2))
        (bdop (build_parabola l1)(build_parabola l2))
)

(defun bdop (coef1 coef2)(cond
   ((and (listp coef1) (listp coef2))
            (print "Пересечение с осью x параболы 1")
      (print (x_intersect coef1))
      (print "Пересечение с осью x параболы 2")
      (print (x_intersect coef2))
      (print "Пересечение парбол (точки если есть)")
      (print (parabolas_intersect coef1 coef2))
      (print "Функция, которая в любой точке X принимает значение, равное среднему
арифметическому значений заданных парабол в этой точке")
      (print (arith_mean coef1 coef2))
         )
         ((listp coef1)
            (print "Пересечение с осью x параболы 1")
      (print (x_intersect coef1))
         )
         ((listp coef2)
      (print "Пересечение с осью x параболы 2")
      (print (x_intersect coef2))
         )

))
(defun build_parabola (set)(cond
   ((= (full_det (caar set) (caadr set) (caaddr set)) 0) "Невозможно построить параболу!")
   ((= (det1 (car set) (cadr set) (caddr set)) 0) "Невозможно построить параболу!")
   (T (list (/ (det1 (car set) (cadr set) (caddr set)) (full_det (caar set) (caadr set) (caaddr set)))
         (/ (det2 (car set) (cadr set) (caddr set)) (full_det (caar set) (caadr set) (caaddr set)))
         (/ (det3 (car set) (cadr set) (caddr set)) (full_det (caar set) (caadr set) (caaddr set)))))
))
```

```
(defun full_det (x1 x2 x3)(* (- x1 x2) (- x2 x3) (- x3 x1)))

(defun det1 (p1 p2 p3)
   (- (* (car p1) (- (cadr p3) (cadr p2)))
   (* (car p2) (- (cadr p1) (cadr p3)))
   (* (car p3) (- (cadr p2) (cadr p1))))
)

(defun det2 (p1 p2 p3)
   (- (* (cadr p1) (- (*(car p3)(car p3)) (*(car p2)(car p2))))
   (* (cadr p2) (- (*(car p1)(car p1)) (*(car p3)(car p3))))
         (* (cadr p3) (- (*(car p2)(car p2)) (*(car p1)(car p1)))))
)

(defun det3 (p1 p2 p3)
   (- (* (cadr p1) (- (*(car p2)(car p2)(car p3)) (*(car p3)(car p3)(car p2))))
   (* (cadr p2) (- (*(car p3)(car p3)(car p1)) (*(car p1)(car p1)(car p3))))
   (* (cadr p3) (- (*(car p1)(car p1)(car p2)) (*(car p2)(car p2)(car p1)))))
)

(defun x_intersect (coef)(x_intrersect_dop (car coef)(cadr coef)(caddr coef)))

(defun x_intrersect_dop (a b c)(cond
   ((< (- (* b b) (* 4 a c)) 0) "Нет пересечений параболы с осью X!")
   ((= (- (* b b) (* 4 a c)) 0) (list (list (/ (- b) (* 2 a)) 0)))
   (T (list (list (/ (+ (- b) (sqrt (- (* b b) (* 4 a c)))) (* 2 a)) 0)(list (/ (- (- b) (sqrt (- (* b b) (* 4 a c))))
(* 2 a)) 0)))
))

(defun parabolas_intersect (coef1 coef2)
         (intersect_search coef1 (- (car coef1) (car coef2)) (- (cadr coef1) (cadr coef2)) (- (caddr
coef1) (caddr coef2)))
)

(defun intersect_search (coef a b c)(cond
   ((and (= a 0) (= b 0) (= c 0)) "Параболы совпадают!")
   ((and (= a 0) (= b 0)) "Параболы не пересекаются!")
   ((= a 0) (list (list (/ (- c) b) (calculate_y coef (/ (- c) b)))))
   (T (discrim_check coef a b c))
))

(defun discrim_check (coef a b c)(cond
   ((< (- (* b b) (* 4 a c)) 0) "Параболы не пересекаются!")
   ((= (- (* b b) (* 4 a c)) 0) (list (list (/ (- b) (* 2 a)) (calculate_y coef (/ (- b) (* 2 a))))))
   (T (list (list (/ (+ (- b) (sqrt (- (* b b) (* 4 a c)))) (* 2 a)) (calculate_y coef (/ (+ (- b) (sqrt (- (* b
b) (* 4 a c)))) (* 2 a))))
```

```
      (list (/ (- (- b) (sqrt (- (* b b) (* 4 a c)))) (* 2 a)) (calculate_y coef (/ (- (- b) (sqrt (- (* b b) (* 4 a c)))) (* 2 a))))))
))

(defun calculate_y (coef x)
        (+ (* (car coef) x x) (* (cadr coef) x) (caddr coef))
)

(defun arith_mean (coef1 coef2)(cond
   ((and (= (+ (car coef1) (car coef2)) 0) (= (+ (cadr coef1) (cadr coef2)) 0)) (list "Y=" (/ (+ (caddr coef1) (caddr coef2)) 2)))
   ((= (+ (car coef1) (car coef2)) 0) (list "Y=" (/ (+ (cadr coef1) (cadr coef2)) 2) "x+" (/ (+ (caddr coef1) (caddr coef2)) 2)))
   (T (list "Y=" (/ (+ (car coef1) (car coef2)) 2) "x^2+" (/ (+ (cadr coef1) (cadr coef2)) 2) "x+" (/ (+ (caddr coef1) (caddr coef2)) 2)))
))

;; Тест 1: Нормальное построение двух парабол - проверка основного функционала
(print "=== Тест 1: Нормальное построение двух парабол ===")
(print "Цель: Проверить корректное построение парабол и их анализ")
(parB '((0 0) (1 1) (2 4)) '((0 1) (1 0) (2 1)))

;; Тест 2: Параболы не пересекаются - проверка обработки непересекающихся кривых
(print "=== Тест 2: Параболы не пересекаются ===")
(print "Цель: Проверить обработку случая непересекающихся парабол")
(parB '((0 2) (1 3) (2 6)) '((0 -1) (1 -2) (2 -5)))

;; Тест 3: Параболы касаются - проверка граничного случая касания
(print "=== Тест 3: Параболы касаются ===")
(print "Цель: Проверить случай близких, но не пересекающихся парабол")
(parB '((0 0) (1 1) (2 4)) '((0 1) (1 2) (2 5)))

;; Тест 4: Вырожденный случай - точки на одной прямой - проверка обработки ошибок
(print "=== Тест 4: Вырожденный случай - точки на одной прямой ===")
(print "Цель: Проверить обработку невозможности построения параболы")
(parB '((0 0) (1 1) (2 2)) '((0 1) (1 2) (2 3)))

;; Тест 5: Параболы совпадают - проверка идентичных кривых
(print "=== Тест 5: Параболы совпадают ===")
(print "Цель: Проверить обработку идентичных парабол")
(parB '((0 0) (1 1) (2 4)) '((0 0) (1 1) (2 4)))

;; Тест 6: Одна точка пересечения - проверка вырожденного пересечения
(print "=== Тест 6: Одна точка пересечения ===")
(print "Цель: Проверить нахождение одной точки пересечения")
(parB '((0 0) (1 1) (2 4)) '((0 0) (1 0) (2 0)))

;; Тест 7: Вертикальные "параболы" - проверка особых случаев построения
```

```
(print "=== Тест 7: Вертикальные параболы ===")
(print "Цель: Проверить обработку вертикальных линий")
(parB '((0 0) (0 1) (0 4)) '((1 0) (1 1) (1 4)))

;; Тест 8: Комплексные корни - проверка парабол без действительных корней
(print "=== Тест 8: Комплексные корни ===")
(print "Цель: Проверить обработку парабол без действительных пересечений с осью X")
(parB '((0 2) (1 3) (2 6)) '((0 1) (1 2) (2 5)))

;; Тест 9: Большие числа - проверка устойчивости к большим значениям
(print "=== Тест 9: Большие числа ===")
(print "Цель: Проверить устойчивость к большим значениям")
(parB '((100 10000) (200 40000) (300 90000)) '((100 5000) (200 10000) (300 15000)))

;; Тест 10: Отрицательные координаты - проверка работы с отрицательными значениями
(print "=== Тест 10: Отрицательные координаты ===")
(print "Цель: Проверить работу с отрицательными координатами")
(parB '((-2 4) (-1 1) (0 0)) '((-2 1) (-1 0) (0 1)))

;; Тест 11: Только одна парабола строится - проверка частичного успеха
(print "=== Тест 11: Только одна парабола строится ===")
(print "Цель: Проверить обработку частично успешного построения")
(parB '((0 0) (1 1) (2 2)) '((0 0) (1 1) (2 4)))

;; Тест 12: Горизонтальные линии - проверка вырожденных парабол
(print "=== Тест 12: Горизонтальные линии ===")
(print "Цель: Проверить построение горизонтальных линий")
(parB '((0 0) (1 0) (2 0)) '((0 1) (1 1) (2 1)))

;; Тест 13: Симметричные параболы - проверка симметричных случаев
(print "=== Тест 13: Симметричные параболы ===")
(print "Цель: Проверить обработку симметричных парабол")
(parB '((-1 1) (0 0) (1 1)) '((-1 2) (0 1) (1 2)))

;; Тест 14: Параболы с разными направлениями - проверка разнонаправленных кривых
(print "=== Тест 14: Параболы с разными направлениями ===")
(print "Цель: Проверить параболы с разной выпуклостью")
(parB '((0 0) (1 1) (2 4)) '((0 4) (1 1) (2 0)))
```

6.Итог пункт C

6.1 Тесты для пункта C

```
;; Тест 1: Нормальные треугольники - проверка основного функционала
(print "=== Тест 1: Нормальные треугольники ===")
(print "Цель: Проверить корректное построение треугольников и их анализ")
(triangleC '((0 0) (3 0) (0 4) (1 1) (2 2)))
```

```
;; Тест 2: Равносторонний треугольник
(print "=== Тест 2: Равносторонний треугольник ===")
(print "Цель: Проверить обнаружение равносторонних треугольников")
(triangleC '((0 0) (2 0) (1 1.732) (1 0) (0 1)))

;; Тест 3: Равнобедренные треугольники
(print "=== Тест 3: Равнобедренные треугольники ===")
(print "Цель: Проверить обнаружение равнобедренных треугольников")
(triangleC '((0 0) (4 0) (2 3) (1 0) (0 2)))

;; Тест 4: Прямоугольные треугольники
(print "=== Тест 4: Прямоугольные треугольники ===")
(print "Цель: Проверить обнаружение прямоугольных треугольников")
(triangleC '((0 0) (3 0) (0 4) (1 0) (0 1)))

;; Тест 5: Подобные треугольники
(print "=== Тест 5: Подобные треугольники ===")
(print "Цель: Проверить обнаружение подобных треугольников")
(triangleC '((0 0) (3 0) (0 4)
             (0 0) (6 0) (0 8)
             (1 1) (4 1) (1 5)))

;; Тест 6: Точки на одной прямой
(print "=== Тест 6: Точки на одной прямой ===")
(print "Цель: Проверить обработку вырожденных случаев")
(triangleC '((0 0) (1 1) (2 2) (3 3) (4 4)))

;; Тест 7: Все точки совпадают
(print "=== Тест 7: Все точки совпадают ===")
(print "Цель: Проверить обработку идентичных точек")
(triangleC '((0 0) (0 0) (0 0) (0 0) (0 0)))

;; Тест 8: Только 3 точки
(print "=== Тест 8: Только 3 точки ===")
(print "Цель: Проверить работу с минимальным количеством точек")
(triangleC '((0 0) (3 0) (0 4)))

;; Тест 9: Большие координаты
(print "=== Тест 9: Большие координаты ===")
(print "Цель: Проверить устойчивость к большим значениям")
(triangleC '((1000 1000) (3000 1000) (1000 4000) (2000 2000) (1500 1500)))

;; Тест 10: Отрицательные координаты
(print "=== Тест 10: Отрицательные координаты ===")
(print "Цель: Проверить работу с отрицательными координатами")
(triangleC '((-2 -2) (1 -2) (-2 1) (0 0) (-1 -1)))
```

```
;; Тест 11: Разные типы треугольников в одном наборе
(print "=== Тест 11: Разные типы треугольников в одном наборе ===")
(print "Цель: Проверить классификацию смешанного набора треугольников")
(triangleC '((0 0) (3 0) (0 4)          ; прямоугольный
          (0 0) (2 0) (1 1.732)      ; равносторонний
          (0 0) (4 0) (2 3)          ; равнобедренный
          (1 1) (2 2) (3 3)))        ; вырожденный

;; Тест 12: Точки образуют только вырожденные треугольники
(print "=== Тест 12: Точки образуют только вырожденные треугольники ===")
(print "Цель: Проверить обработку когда все треугольники вырожденные")
(triangleC '((0 0) (1 1) (2 2) (3 3) (0 1) (1 2)))

;; Тест 13: Точки в вершинах квадрата
(print "=== Тест 13: Точки в вершинах квадрата ===")
(print "Цель: Проверить построение треугольников из точек квадрата")
(triangleC '((0 0) (2 0) (2 2) (0 2) (1 1)))

;; Тест 14: Много точек с дубликатами
(print "=== Тест 14: Много точек с дубликатами ===")
(print "Цель: Проверить обработку дублирующихся точек")
(triangleC '((0 0) (3 0) (0 4) (0 0) (3 0) (0 4) (1 1) (1 1)))

;; Тест 15: Комплексный тест со всеми типами
(print "=== Тест 15: Комплексный тест со всеми типами ===")
(print "Цель: Комплексная проверка всех функций анализа треугольников")
(triangleC '((0 0) (4 0) (0 3)          ; прямоугольный
          (5 0) (7 0) (6 1.732)      ; равносторонний
          (8 0) (12 0) (10 4)        ; равнобедренный
          (0 5) (3 5) (0 9)          ; прямоугольный (подобный первому)
          (1 1) (2 2) (3 3)))        ; вырожденный
```

6.2 Ответы на тесты

```
"=== Тест 1: Нормальные треугольники ==="
"Цель: Проверить корректное построение треугольников и их анализ"
"Пункт C"
"Треугольники:"
(((0 0) (3 0) (0 4)) ((0 0) (3 0) (1 1)) ((0 0) (3 0) (2 2))
 ((0 0) (0 4) (1 1)) ((0 0) (0 4) (2 2)) ((3 0) (0 4) (1 1))
 ((3 0) (0 4) (2 2)) ((3 0) (1 1) (2 2)) ((0 4) (1 1) (2 2)))
"Равнобедренные:"
(((0 0) (0 4) (2 2)) ((3 0) (1 1) (2 2)))
"Равносторонние:"
NIL
"Прямоугольные:"
(((0 0) (3 0) (0 4)) ((0 0) (0 4) (2 2)) ((0 4) (1 1) (2 2)))
```

```
"Подобные:"
NIL
"=== Тест 2: Равносторонний треугольник ==="
"Цель: Проверить обнаружение равносторонних треугольников"
"Пункт C"
"Треугольники:"
(((0 0) (2 0) (1 1.732)) ((0 0) (2 0) (0 1)) ((0 0) (1 1.732) (1
0))
 ((0 0) (1 1.732) (0 1)) ((0 0) (1 0) (0 1)) ((2 0) (1 1.732) (1
0))
 ((2 0) (1 1.732) (0 1)) ((2 0) (1 0) (0 1)) ((1 1.732) (1 0) (0
1)))
"Равнобедренные:"
(((0 0) (2 0) (1 1.732)) ((0 0) (1 0) (0 1)))
"Равносторонние:"
(((0 0) (2 0) (1 1.732)))
"Прямоугольные:"
(((0 0) (2 0) (0 1)) ((0 0) (1 1.732) (1 0)) ((0 0) (1 0) (0 1))
 ((2 0) (1 1.732) (1 0)))
"Подобные:"
((((0 0) (1 1.732) (1 0)) ((2 0) (1 1.732) (1 0))))
"=== Тест 3: Равнобедренные треугольники ==="
"Цель: Проверить обнаружение равнобедренных треугольников"
"Пункт C"
"Треугольники:"
(((0 0) (4 0) (2 3)) ((0 0) (4 0) (0 2)) ((0 0) (2 3) (1 0))
 ((0 0) (2 3) (0 2)) ((0 0) (1 0) (0 2)) ((4 0) (2 3) (1 0))
 ((4 0) (2 3) (0 2)) ((4 0) (1 0) (0 2)) ((2 3) (1 0) (0 2)))
"Равнобедренные:"
(((0 0) (4 0) (2 3)) ((2 3) (1 0) (0 2)))
"Равносторонние:"
NIL
"Прямоугольные:"
(((0 0) (4 0) (0 2)) ((0 0) (1 0) (0 2)) ((2 3) (1 0) (0 2)))
"Подобные:"
((((0 0) (4 0) (0 2)) ((0 0) (1 0) (0 2))))
"=== Тест 4: Прямоугольные треугольники ==="
"Цель: Проверить обнаружение прямоугольных треугольников"
"Пункт C"
"Треугольники:"
(((0 0) (3 0) (0 4)) ((0 0) (3 0) (0 1)) ((0 0) (0 4) (1 0))
 ((0 0) (1 0) (0 1)) ((3 0) (0 4) (1 0)) ((3 0) (0 4) (0 1))
 ((3 0) (1 0) (0 1)) ((0 4) (1 0) (0 1)))
"Равнобедренные:"
(((0 0) (1 0) (0 1)))
"Равносторонние:"
NIL
"Прямоугольные:"
(((0 0) (3 0) (0 4)) ((0 0) (3 0) (0 1)) ((0 0) (0 4) (1 0))
 ((0 0) (1 0) (0 1)))
"Подобные:"
```

```
NIL
"=== Тест 5: Подобные треугольники ==="
"Цель: Проверить обнаружение подобных треугольников"
"Пункт C"
"Треугольники:"
(((3 0) (0 4) (0 0)) ((3 0) (0 4) (6 0)) ((3 0) (0 4) (0 8))
 ((3 0) (0 4) (1 1)) ((3 0) (0 4) (4 1)) ((3 0) (0 4) (1 5))
 ((3 0) (0 0) (0 8)) ((3 0) (0 0) (1 1)) ((3 0) (0 0) (4 1))
 ((3 0) (0 0) (1 5)) ((3 0) (6 0) (0 8)) ((3 0) (6 0) (1 1))
 ((3 0) (6 0) (4 1)) ((3 0) (6 0) (1 5)) ((3 0) (0 8) (1 1))
 ((3 0) (0 8) (4 1)) ((3 0) (0 8) (1 5)) ((3 0) (1 1) (4 1))
 ((3 0) (1 1) (1 5)) ((3 0) (4 1) (1 5)) ((0 4) (0 0) (6 0))
 ((0 4) (0 0) (1 1)) ((0 4) (0 0) (4 1)) ((0 4) (0 0) (1 5))
 ((0 4) (6 0) (0 8)) ((0 4) (6 0) (1 1)) ((0 4) (6 0) (4 1))
 ((0 4) (6 0) (1 5)) ((0 4) (0 8) (1 1)) ((0 4) (0 8) (4 1))
 ((0 4) (0 8) (1 5)) ((0 4) (1 1) (4 1)) ((0 4) (1 1) (1 5))
 ((0 4) (4 1) (1 5)) ((0 0) (6 0) (0 8)) ((0 0) (6 0) (1 1))
 ((0 0) (6 0) (4 1)) ((0 0) (6 0) (1 5)) ((0 0) (0 8) (1 1))
 ((0 0) (0 8) (4 1)) ((0 0) (0 8) (1 5)) ((0 0) (1 1) (4 1))
 ((0 0) (1 1) (1 5)) ((0 0) (4 1) (1 5)) ((6 0) (0 8) (1 1))
 ((6 0) (0 8) (4 1)) ((6 0) (0 8) (1 5)) ((6 0) (1 1) (4 1))
 ((6 0) (1 1) (1 5)) ((6 0) (4 1) (1 5)) ((0 8) (1 1) (4 1))
 ((0 8) (1 1) (1 5)) ((0 8) (4 1) (1 5)) ((1 1) (4 1) (1 5)))
"Равнобедренные:"
(((3 0) (0 4) (4 1)) ((0 4) (4 1) (1 5)))
"Равносторонние:"
NIL
"Прямоугольные:"
(((3 0) (0 4) (0 0)) ((3 0) (0 0) (0 8)) ((0 4) (0 0) (6 0))
 ((0 4) (6 0) (1 5)) ((0 0) (6 0) (0 8)) ((1 1) (4 1) (1 5)))
"Подобные:"
((((3 0) (0 4) (0 0)) ((0 0) (6 0) (0 8)))
 (((3 0) (0 4) (0 0)) ((1 1) (4 1) (1 5)))
 (((3 0) (0 4) (4 1)) ((0 4) (4 1) (1 5)))
 (((3 0) (0 4) (1 5)) ((3 0) (4 1) (1 5)))
 (((3 0) (0 0) (1 1)) ((3 0) (6 0) (4 1)))
 (((3 0) (0 0) (1 1)) ((3 0) (1 1) (4 1)))
 (((3 0) (0 0) (4 1)) ((0 0) (1 1) (4 1)))
 (((3 0) (6 0) (1 1)) ((6 0) (1 1) (4 1)))
 (((3 0) (6 0) (4 1)) ((3 0) (1 1) (4 1)))
 (((0 4) (0 0) (1 1)) ((0 4) (0 8) (1 5)))
 (((0 4) (0 0) (1 1)) ((0 4) (1 1) (1 5)))
 (((0 4) (0 0) (1 5)) ((0 0) (1 1) (1 5)))
 (((0 4) (0 8) (1 1)) ((0 8) (1 1) (1 5)))
 (((0 4) (0 8) (1 5)) ((0 4) (1 1) (1 5)))
 (((0 0) (6 0) (0 8)) ((1 1) (4 1) (1 5)))
 (((6 0) (0 8) (1 5)) ((6 0) (4 1) (1 5))))
"=== Тест 6: Точки на одной прямой ==="
"Цель: Проверить обработку вырожденных случаев"
"Пункт C"
"Треугольники:"
```

```
NIL
"Равнобедренные:"
NIL
"Равносторонние:"
NIL
"Прямоугольные:"
NIL
"Подобные:"
NIL
"=== Тест 7: Все точки совпадают ==="
"Цель: Проверить обработку идентичных точек"
"Пункт C"
"Треугольники:"
NIL
"Равнобедренные:"
NIL
"Равносторонние:"
NIL
"Прямоугольные:"
NIL
"Подобные:"
NIL
"=== Тест 8: Только 3 точки ==="
"Цель: Проверить работу с минимальным количеством точек"
"Пункт C"
"Треугольники:"
(((0 0) (3 0) (0 4)))
"Равнобедренные:"
NIL
"Равносторонние:"
NIL
"Прямоугольные:"
(((0 0) (3 0) (0 4)))
"Подобные:"
NIL
"=== Тест 9: Большие координаты ==="
"Цель: Проверить устойчивость к большим значениям"
"Пункт C"
"Треугольники:"
(((1000 1000) (3000 1000) (1000 4000)) ((1000 1000) (3000 1000)
(2000 2000))
 ((1000 1000) (3000 1000) (1500 1500)) ((1000 1000) (1000 4000)
(2000 2000))
 ((1000 1000) (1000 4000) (1500 1500)) ((3000 1000) (1000 4000)
(2000 2000))
 ((3000 1000) (1000 4000) (1500 1500)) ((3000 1000) (2000 2000)
(1500 1500))
 ((1000 4000) (2000 2000) (1500 1500)))
"Равнобедренные:"
(((1000 1000) (3000 1000) (2000 2000)))
"Равносторонние:"
```

```
NIL
"Прямоугольные:"
(((1000 1000) (3000 1000) (1000 4000)))
"Подобные:"
NIL
"=== Тест 10: Отрицательные координаты ==="
"Цель: Проверить работу с отрицательными координатами"
"Пункт C"
"Треугольники:"
(((-2 -2) (1 -2) (-2 1)) ((-2 -2) (1 -2) (0 0)) ((-2 -2) (1 -2)
(-1 -1))
 ((-2 -2) (-2 1) (0 0)) ((-2 -2) (-2 1) (-1 -1)) ((1 -2) (-2 1) (0
0))
 ((1 -2) (-2 1) (-1 -1)) ((1 -2) (0 0) (-1 -1)) ((-2 1) (0 0) (-1
-1)))
"Равнобедренные:"
(((-2 -2) (1 -2) (-2 1)) ((1 -2) (-2 1) (0 0)) ((1 -2) (-2 1) (-1
-1))
 ((1 -2) (0 0) (-1 -1)) ((-2 1) (0 0) (-1 -1)))
"Равносторонние:"
NIL
"Прямоугольные:"
(((-2 -2) (1 -2) (-2 1)))
"Подобные:"
((((-2 -2) (1 -2) (0 0)) ((-2 -2) (-2 1) (0 0)))
 (((-2 -2) (1 -2) (-1 -1)) ((-2 -2) (-2 1) (-1 -1)))
 (((1 -2) (-2 1) (0 0)) ((1 -2) (-2 1) (-1 -1)))
 (((1 -2) (0 0) (-1 -1)) ((-2 1) (0 0) (-1 -1))))
"=== Тест 11: Разные типы треугольников в одном наборе ==="
"Цель: Проверить классификацию смешанного набора треугольников"
"Пункт C"
"Треугольники:"
(((3 0) (0 4) (2 0)) ((3 0) (0 4) (1 1.732)) ((3 0) (0 4) (0 0))
 ((3 0) (0 4) (4 0)) ((3 0) (0 4) (2 3)) ((3 0) (0 4) (1 1))
 ((3 0) (0 4) (2 2)) ((3 0) (0 4) (3 3)) ((3 0) (2 0) (1 1.732))
 ((3 0) (2 0) (2 3)) ((3 0) (2 0) (1 1)) ((3 0) (2 0) (2 2))
 ((3 0) (2 0) (3 3)) ((3 0) (1 1.732) (0 0)) ((3 0) (1 1.732) (4
0))
 ((3 0) (1 1.732) (2 3)) ((3 0) (1 1.732) (1 1)) ((3 0) (1 1.732)
(2 2))
 ((3 0) (1 1.732) (3 3)) ((3 0) (0 0) (2 3)) ((3 0) (0 0) (1 1))
 ((3 0) (0 0) (2 2)) ((3 0) (0 0) (3 3)) ((3 0) (4 0) (2 3))
 ((3 0) (4 0) (1 1)) ((3 0) (4 0) (2 2)) ((3 0) (4 0) (3 3))
 ((3 0) (2 3) (1 1)) ((3 0) (2 3) (2 2)) ((3 0) (2 3) (3 3))
 ((3 0) (1 1) (2 2)) ((3 0) (1 1) (3 3)) ((3 0) (2 2) (3 3))
 ((0 4) (2 0) (1 1.732)) ((0 4) (2 0) (0 0)) ((0 4) (2 0) (4 0))
 ((0 4) (2 0) (2 3)) ((0 4) (2 0) (1 1)) ((0 4) (2 0) (2 2))
 ((0 4) (2 0) (3 3)) ((0 4) (1 1.732) (0 0)) ((0 4) (1 1.732) (4
0))
 ((0 4) (1 1.732) (2 3)) ((0 4) (1 1.732) (1 1)) ((0 4) (1 1.732)
(2 2))
```

```
 ((0 4) (1 1.732) (3 3)) ((0 4) (0 0) (4 0)) ((0 4) (0 0) (2 3))
 ((0 4) (0 0) (1 1)) ((0 4) (0 0) (2 2)) ((0 4) (0 0) (3 3))
 ((0 4) (4 0) (2 3)) ((0 4) (4 0) (1 1)) ((0 4) (4 0) (3 3))
 ((0 4) (2 3) (1 1)) ((0 4) (2 3) (2 2)) ((0 4) (2 3) (3 3))
 ((0 4) (1 1) (2 2)) ((0 4) (1 1) (3 3)) ((0 4) (2 2) (3 3))
 ((2 0) (1 1.732) (0 0)) ((2 0) (1 1.732) (4 0)) ((2 0) (1 1.732)
(2 3))
 ((2 0) (1 1.732) (1 1)) ((2 0) (1 1.732) (2 2)) ((2 0) (1 1.732)
(3 3))
 ((2 0) (0 0) (2 3)) ((2 0) (0 0) (1 1)) ((2 0) (0 0) (2 2))
 ((2 0) (0 0) (3 3)) ((2 0) (4 0) (2 3)) ((2 0) (4 0) (1 1))
 ((2 0) (4 0) (2 2)) ((2 0) (4 0) (3 3)) ((2 0) (2 3) (1 1))
 ((2 0) (2 3) (3 3)) ((2 0) (1 1) (2 2)) ((2 0) (1 1) (3 3))
 ((2 0) (2 2) (3 3)) ((1 1.732) (0 0) (4 0)) ((1 1.732) (0 0) (2
3))
 ((1 1.732) (0 0) (1 1)) ((1 1.732) (0 0) (2 2)) ((1 1.732) (0 0)
(3 3))
 ((1 1.732) (4 0) (2 3)) ((1 1.732) (4 0) (1 1)) ((1 1.732) (4 0)
(2 2))
 ((1 1.732) (4 0) (3 3)) ((1 1.732) (2 3) (1 1)) ((1 1.732) (2 3)
(2 2))
 ((1 1.732) (2 3) (3 3)) ((1 1.732) (1 1) (2 2)) ((1 1.732) (1 1)
(3 3))
 ((1 1.732) (2 2) (3 3)) ((0 0) (4 0) (2 3)) ((0 0) (4 0) (1 1))
 ((0 0) (4 0) (2 2)) ((0 0) (4 0) (3 3)) ((0 0) (2 3) (1 1))
 ((0 0) (2 3) (2 2)) ((0 0) (2 3) (3 3)) ((4 0) (2 3) (1 1))
 ((4 0) (2 3) (2 2)) ((4 0) (2 3) (3 3)) ((4 0) (1 1) (2 2))
 ((4 0) (1 1) (3 3)) ((4 0) (2 2) (3 3)) ((2 3) (1 1) (2 2))
 ((2 3) (1 1) (3 3)) ((2 3) (2 2) (3 3)))
"Равнобедренные:"
(((3 0) (0 0) (3 3)) ((3 0) (2 3) (1 1)) ((3 0) (1 1) (2 2))
 ((0 4) (2 0) (3 3)) ((0 4) (0 0) (4 0)) ((0 4) (0 0) (2 2))
 ((0 4) (4 0) (1 1)) ((0 4) (4 0) (3 3)) ((0 4) (2 3) (1 1))
 ((0 4) (1 1) (3 3)) ((2 0) (1 1.732) (0 0)) ((2 0) (1 1.732) (4
0))
 ((2 0) (1 1.732) (2 2)) ((2 0) (0 0) (1 1)) ((2 0) (0 0) (2 2))
 ((2 0) (4 0) (2 2)) ((2 0) (4 0) (3 3)) ((2 0) (1 1) (2 2))
 ((0 0) (4 0) (2 3)) ((0 0) (4 0) (2 2)) ((4 0) (1 1) (3 3))
 ((2 3) (2 2) (3 3)))
"Равносторонние:"
(((2 0) (1 1.732) (0 0)))
"Прямоугольные:"
(((3 0) (0 4) (0 0)) ((3 0) (2 0) (2 3)) ((3 0) (2 0) (2 2))
 ((3 0) (2 0) (3 3)) ((3 0) (0 0) (3 3)) ((3 0) (4 0) (3 3))
 ((3 0) (2 3) (1 1)) ((3 0) (2 3) (3 3)) ((0 4) (2 0) (0 0))
 ((0 4) (2 0) (3 3)) ((0 4) (0 0) (4 0)) ((0 4) (0 0) (2 2))
 ((0 4) (2 3) (1 1)) ((0 4) (1 1) (2 2)) ((0 4) (2 2) (3 3))
 ((2 0) (0 0) (2 3)) ((2 0) (0 0) (1 1)) ((2 0) (0 0) (2 2))
 ((2 0) (4 0) (2 3)) ((2 0) (4 0) (2 2)) ((2 0) (2 3) (3 3))
 ((2 0) (1 1) (2 2)) ((2 0) (1 1) (3 3)) ((1 1.732) (0 0) (4 0))
 ((0 0) (4 0) (2 2)) ((4 0) (1 1) (2 2)) ((4 0) (2 2) (3 3))
```

```
 ((2 3) (2 2) (3 3)))
"Подобные:"
((((3 0) (0 4) (2 3)) ((3 0) (0 4) (1 1)))
 (((3 0) (0 4) (2 3)) ((3 0) (2 0) (1 1)))
 (((3 0) (0 4) (2 3)) ((0 4) (2 0) (2 2)))
 (((3 0) (0 4) (2 3)) ((2 0) (4 0) (1 1)))
 (((3 0) (0 4) (2 3)) ((2 0) (2 2) (3 3)))
 (((3 0) (0 4) (2 3)) ((2 3) (1 1) (2 2)))
 (((3 0) (0 4) (1 1)) ((3 0) (2 0) (1 1)))
 (((3 0) (0 4) (1 1)) ((0 4) (2 0) (2 2)))
 (((3 0) (0 4) (1 1)) ((2 0) (4 0) (1 1)))
 (((3 0) (0 4) (1 1)) ((2 0) (2 2) (3 3)))
 (((3 0) (0 4) (1 1)) ((2 3) (1 1) (2 2)))
 (((3 0) (2 0) (2 3)) ((3 0) (2 0) (3 3)))
 (((3 0) (2 0) (2 3)) ((3 0) (4 0) (3 3)))
 (((3 0) (2 0) (2 3)) ((3 0) (2 3) (3 3)))
 (((3 0) (2 0) (2 3)) ((2 0) (2 3) (3 3)))
 (((3 0) (2 0) (1 1)) ((0 4) (2 0) (2 2)))
 (((3 0) (2 0) (1 1)) ((2 0) (4 0) (1 1)))
 (((3 0) (2 0) (1 1)) ((2 0) (2 2) (3 3)))
 (((3 0) (2 0) (1 1)) ((2 3) (1 1) (2 2)))
 (((3 0) (2 0) (2 2)) ((0 4) (2 0) (0 0)))
 (((3 0) (2 0) (2 2)) ((0 4) (1 1) (2 2)))
 (((3 0) (2 0) (2 2)) ((0 4) (2 2) (3 3)))
 (((3 0) (2 0) (2 2)) ((2 0) (1 1) (3 3)))
 (((3 0) (2 0) (2 2)) ((4 0) (1 1) (2 2)))
 (((3 0) (2 0) (2 2)) ((4 0) (2 2) (3 3)))
 (((3 0) (2 0) (3 3)) ((3 0) (4 0) (3 3)))
 (((3 0) (2 0) (3 3)) ((3 0) (2 3) (3 3)))
 (((3 0) (2 0) (3 3)) ((2 0) (2 3) (3 3)))
 (((3 0) (0 0) (1 1)) ((3 0) (2 2) (3 3)))
 (((3 0) (0 0) (1 1)) ((2 0) (0 0) (3 3)))
 (((3 0) (0 0) (1 1)) ((2 0) (2 3) (1 1)))
 (((3 0) (0 0) (2 2)) ((3 0) (1 1) (3 3)))
 (((3 0) (0 0) (2 2)) ((0 4) (0 0) (3 3)))
 (((3 0) (0 0) (2 2)) ((0 0) (4 0) (3 3)))
 (((3 0) (0 0) (3 3)) ((3 0) (2 3) (1 1)))
 (((3 0) (0 0) (3 3)) ((0 4) (2 0) (3 3)))
 (((3 0) (0 0) (3 3)) ((0 4) (0 0) (4 0)))
 (((3 0) (0 0) (3 3)) ((0 4) (0 0) (2 2)))
 (((3 0) (0 0) (3 3)) ((0 4) (2 3) (1 1)))
 (((3 0) (0 0) (3 3)) ((2 0) (0 0) (1 1)))
 (((3 0) (0 0) (3 3)) ((2 0) (0 0) (2 2)))
 (((3 0) (0 0) (3 3)) ((2 0) (4 0) (2 2)))
 (((3 0) (0 0) (3 3)) ((2 0) (1 1) (2 2)))
 (((3 0) (0 0) (3 3)) ((0 0) (4 0) (2 2)))
 (((3 0) (0 0) (3 3)) ((2 3) (2 2) (3 3)))
 (((3 0) (4 0) (2 3)) ((4 0) (2 3) (3 3)))
 (((3 0) (4 0) (1 1)) ((3 0) (2 3) (2 2)))
 (((3 0) (4 0) (1 1)) ((0 4) (2 0) (1 1)))
 (((3 0) (4 0) (1 1)) ((0 4) (2 3) (3 3)))
```

```
(((3 0) (4 0) (2 2)) ((0 4) (2 0) (4 0)))
(((3 0) (4 0) (2 2)) ((0 4) (0 0) (1 1)))
(((3 0) (4 0) (2 2)) ((0 4) (2 3) (2 2)))
(((3 0) (4 0) (2 2)) ((0 0) (4 0) (1 1)))
(((3 0) (4 0) (2 2)) ((2 3) (1 1) (3 3)))
(((3 0) (4 0) (3 3)) ((3 0) (2 3) (3 3)))
(((3 0) (4 0) (3 3)) ((2 0) (2 3) (3 3)))
(((3 0) (2 3) (1 1)) ((0 4) (2 0) (3 3)))
(((3 0) (2 3) (1 1)) ((0 4) (0 0) (4 0)))
(((3 0) (2 3) (1 1)) ((0 4) (0 0) (2 2)))
(((3 0) (2 3) (1 1)) ((0 4) (2 3) (1 1)))
(((3 0) (2 3) (1 1)) ((2 0) (0 0) (1 1)))
(((3 0) (2 3) (1 1)) ((2 0) (0 0) (2 2)))
(((3 0) (2 3) (1 1)) ((2 0) (4 0) (2 2)))
(((3 0) (2 3) (1 1)) ((2 0) (1 1) (2 2)))
(((3 0) (2 3) (1 1)) ((0 0) (4 0) (2 2)))
(((3 0) (2 3) (1 1)) ((2 3) (2 2) (3 3)))
(((3 0) (2 3) (2 2)) ((0 4) (2 0) (1 1)))
(((3 0) (2 3) (2 2)) ((0 4) (2 3) (3 3)))
(((3 0) (2 3) (3 3)) ((2 0) (2 3) (3 3)))
(((3 0) (1 1) (2 2)) ((2 0) (4 0) (3 3)))
(((3 0) (1 1) (3 3)) ((0 4) (0 0) (3 3)))
(((3 0) (1 1) (3 3)) ((0 0) (4 0) (3 3)))
(((3 0) (2 2) (3 3)) ((2 0) (0 0) (3 3)))
(((3 0) (2 2) (3 3)) ((2 0) (2 3) (1 1)))
(((0 4) (2 0) (0 0)) ((0 4) (1 1) (2 2)))
(((0 4) (2 0) (0 0)) ((0 4) (2 2) (3 3)))
(((0 4) (2 0) (0 0)) ((2 0) (1 1) (3 3)))
(((0 4) (2 0) (0 0)) ((4 0) (1 1) (2 2)))
(((0 4) (2 0) (0 0)) ((4 0) (2 2) (3 3)))
(((0 4) (2 0) (4 0)) ((0 4) (0 0) (1 1)))
(((0 4) (2 0) (4 0)) ((0 4) (2 3) (2 2)))
(((0 4) (2 0) (4 0)) ((0 0) (4 0) (1 1)))
(((0 4) (2 0) (4 0)) ((2 3) (1 1) (3 3)))
(((0 4) (2 0) (1 1)) ((0 4) (2 3) (3 3)))
(((0 4) (2 0) (2 2)) ((2 0) (4 0) (1 1)))
(((0 4) (2 0) (2 2)) ((2 0) (2 2) (3 3)))
(((0 4) (2 0) (2 2)) ((2 3) (1 1) (2 2)))
(((0 4) (2 0) (3 3)) ((0 4) (0 0) (4 0)))
(((0 4) (2 0) (3 3)) ((0 4) (0 0) (2 2)))
(((0 4) (2 0) (3 3)) ((0 4) (2 3) (1 1)))
(((0 4) (2 0) (3 3)) ((2 0) (0 0) (1 1)))
(((0 4) (2 0) (3 3)) ((2 0) (0 0) (2 2)))
(((0 4) (2 0) (3 3)) ((2 0) (4 0) (2 2)))
(((0 4) (2 0) (3 3)) ((2 0) (1 1) (2 2)))
(((0 4) (2 0) (3 3)) ((0 0) (4 0) (2 2)))
(((0 4) (2 0) (3 3)) ((2 3) (2 2) (3 3)))
(((0 4) (0 0) (4 0)) ((0 4) (0 0) (2 2)))
(((0 4) (0 0) (4 0)) ((0 4) (2 3) (1 1)))
(((0 4) (0 0) (4 0)) ((2 0) (0 0) (1 1)))
(((0 4) (0 0) (4 0)) ((2 0) (0 0) (2 2)))
```

```
(((0 4) (0 0) (4 0)) ((2 0) (4 0) (2 2)))
(((0 4) (0 0) (4 0)) ((2 0) (1 1) (2 2)))
(((0 4) (0 0) (4 0)) ((0 0) (4 0) (2 2)))
(((0 4) (0 0) (4 0)) ((2 3) (2 2) (3 3)))
(((0 4) (0 0) (1 1)) ((0 4) (2 3) (2 2)))
(((0 4) (0 0) (1 1)) ((0 0) (4 0) (1 1)))
(((0 4) (0 0) (1 1)) ((2 3) (1 1) (3 3)))
(((0 4) (0 0) (2 2)) ((0 4) (2 3) (1 1)))
(((0 4) (0 0) (2 2)) ((2 0) (0 0) (1 1)))
(((0 4) (0 0) (2 2)) ((2 0) (0 0) (2 2)))
(((0 4) (0 0) (2 2)) ((2 0) (4 0) (2 2)))
(((0 4) (0 0) (2 2)) ((2 0) (1 1) (2 2)))
(((0 4) (0 0) (2 2)) ((0 0) (4 0) (2 2)))
(((0 4) (0 0) (2 2)) ((2 3) (2 2) (3 3)))
(((0 4) (0 0) (3 3)) ((0 0) (4 0) (3 3)))
(((0 4) (4 0) (1 1)) ((0 4) (4 0) (3 3)))
(((0 4) (2 3) (1 1)) ((2 0) (0 0) (1 1)))
(((0 4) (2 3) (1 1)) ((2 0) (0 0) (2 2)))
(((0 4) (2 3) (1 1)) ((2 0) (4 0) (2 2)))
(((0 4) (2 3) (1 1)) ((2 0) (1 1) (2 2)))
(((0 4) (2 3) (1 1)) ((0 0) (4 0) (2 2)))
(((0 4) (2 3) (1 1)) ((2 3) (2 2) (3 3)))
(((0 4) (2 3) (2 2)) ((0 0) (4 0) (1 1)))
(((0 4) (2 3) (2 2)) ((2 3) (1 1) (3 3)))
(((0 4) (1 1) (2 2)) ((0 4) (2 2) (3 3)))
(((0 4) (1 1) (2 2)) ((2 0) (1 1) (3 3)))
(((0 4) (1 1) (2 2)) ((4 0) (1 1) (2 2)))
(((0 4) (1 1) (2 2)) ((4 0) (2 2) (3 3)))
(((0 4) (1 1) (3 3)) ((4 0) (1 1) (3 3)))
(((0 4) (2 2) (3 3)) ((2 0) (1 1) (3 3)))
(((0 4) (2 2) (3 3)) ((4 0) (1 1) (2 2)))
(((0 4) (2 2) (3 3)) ((4 0) (2 2) (3 3)))
(((2 0) (1 1.732) (1 1)) ((1 1.732) (0 0) (1 1)))
(((2 0) (1 1.732) (1 1)) ((1 1.732) (0 0) (2 2)))
(((2 0) (0 0) (2 3)) ((2 0) (4 0) (2 3)))
(((2 0) (0 0) (1 1)) ((2 0) (0 0) (2 2)))
(((2 0) (0 0) (1 1)) ((2 0) (4 0) (2 2)))
(((2 0) (0 0) (1 1)) ((2 0) (1 1) (2 2)))
(((2 0) (0 0) (1 1)) ((0 0) (4 0) (2 2)))
(((2 0) (0 0) (1 1)) ((2 3) (2 2) (3 3)))
(((2 0) (0 0) (2 2)) ((2 0) (4 0) (2 2)))
(((2 0) (0 0) (2 2)) ((2 0) (1 1) (2 2)))
(((2 0) (0 0) (2 2)) ((0 0) (4 0) (2 2)))
(((2 0) (0 0) (2 2)) ((2 3) (2 2) (3 3)))
(((2 0) (0 0) (3 3)) ((2 0) (2 3) (1 1)))
(((2 0) (4 0) (1 1)) ((2 0) (2 2) (3 3)))
(((2 0) (4 0) (1 1)) ((2 3) (1 1) (2 2)))
(((2 0) (4 0) (2 2)) ((2 0) (1 1) (2 2)))
(((2 0) (4 0) (2 2)) ((0 0) (4 0) (2 2)))
(((2 0) (4 0) (2 2)) ((2 3) (2 2) (3 3)))
(((2 0) (1 1) (2 2)) ((0 0) (4 0) (2 2)))
```

```
 (((2 0) (1 1) (2 2)) ((2 3) (2 2) (3 3)))
 (((2 0) (1 1) (3 3)) ((4 0) (1 1) (2 2)))
 (((2 0) (1 1) (3 3)) ((4 0) (2 2) (3 3)))
 (((2 0) (2 2) (3 3)) ((2 3) (1 1) (2 2)))
 (((1 1.732) (0 0) (1 1)) ((1 1.732) (0 0) (2 2)))
 (((0 0) (4 0) (1 1)) ((2 3) (1 1) (3 3)))
 (((0 0) (4 0) (2 2)) ((2 3) (2 2) (3 3)))
 (((0 0) (2 3) (2 2)) ((4 0) (2 3) (2 2)))
 (((4 0) (1 1) (2 2)) ((4 0) (2 2) (3 3))))
"=== Тест 12: Точки образуют только вырожденные треугольники ==="
"Цель: Проверить обработку когда все треугольники вырожденные"
"Пункт C"
"Треугольники:"
(((0 0) (1 1) (0 1)) ((0 0) (1 1) (1 2)) ((0 0) (2 2) (0 1))
 ((0 0) (2 2) (1 2)) ((0 0) (3 3) (0 1)) ((0 0) (3 3) (1 2))
 ((0 0) (0 1) (1 2)) ((1 1) (2 2) (0 1)) ((1 1) (2 2) (1 2))
 ((1 1) (3 3) (0 1)) ((1 1) (3 3) (1 2)) ((1 1) (0 1) (1 2))
 ((2 2) (3 3) (0 1)) ((2 2) (3 3) (1 2)) ((2 2) (0 1) (1 2))
 ((3 3) (0 1) (1 2)))
"Равнобедренные:"
(((0 0) (1 1) (0 1)) ((0 0) (3 3) (1 2)) ((1 1) (2 2) (1 2))
 ((1 1) (0 1) (1 2)))
"Равносторонние:"
NIL
"Прямоугольные:"
(((0 0) (1 1) (0 1)) ((1 1) (2 2) (1 2)) ((1 1) (0 1) (1 2)))
"Подобные:"
((((0 0) (1 1) (0 1)) ((1 1) (2 2) (1 2)))
 (((0 0) (1 1) (0 1)) ((1 1) (0 1) (1 2)))
 (((0 0) (1 1) (1 2)) ((0 0) (0 1) (1 2)))
 (((0 0) (1 1) (1 2)) ((1 1) (2 2) (0 1)))
 (((0 0) (1 1) (1 2)) ((2 2) (3 3) (1 2)))
 (((0 0) (1 1) (1 2)) ((2 2) (0 1) (1 2)))
 (((0 0) (2 2) (0 1)) ((0 0) (2 2) (1 2)))
 (((0 0) (2 2) (0 1)) ((1 1) (3 3) (1 2)))
 (((0 0) (2 2) (1 2)) ((1 1) (3 3) (1 2)))
 (((0 0) (0 1) (1 2)) ((1 1) (2 2) (0 1)))
 (((0 0) (0 1) (1 2)) ((2 2) (3 3) (1 2)))
 (((0 0) (0 1) (1 2)) ((2 2) (0 1) (1 2)))
 (((1 1) (2 2) (0 1)) ((2 2) (3 3) (1 2)))
 (((1 1) (2 2) (0 1)) ((2 2) (0 1) (1 2)))
 (((1 1) (2 2) (1 2)) ((1 1) (0 1) (1 2)))
 (((2 2) (3 3) (0 1)) ((3 3) (0 1) (1 2)))
 (((2 2) (3 3) (1 2)) ((2 2) (0 1) (1 2))))
"=== Тест 13: Точки в вершинах квадрата ==="
"Цель: Проверить построение треугольников из точек квадрата"
"Пункт C"
"Треугольники:"
(((0 0) (2 0) (2 2)) ((0 0) (2 0) (0 2)) ((0 0) (2 0) (1 1))
 ((0 0) (2 2) (0 2)) ((0 0) (0 2) (1 1)) ((2 0) (2 2) (0 2))
 ((2 0) (2 2) (1 1)) ((2 2) (0 2) (1 1)))
```

```
"Равнобедренные:"
(((0 0) (2 0) (2 2)) ((0 0) (2 0) (0 2)) ((0 0) (2 0) (1 1))
 ((0 0) (2 2) (0 2)) ((0 0) (0 2) (1 1)) ((2 0) (2 2) (0 2))
 ((2 0) (2 2) (1 1)) ((2 2) (0 2) (1 1)))
"Равносторонние:"
NIL
"Прямоугольные:"
(((0 0) (2 0) (2 2)) ((0 0) (2 0) (0 2)) ((0 0) (2 0) (1 1))
 ((0 0) (2 2) (0 2)) ((0 0) (0 2) (1 1)) ((2 0) (2 2) (0 2))
 ((2 0) (2 2) (1 1)) ((2 2) (0 2) (1 1)))
"Подобные:"
((((0 0) (2 0) (2 2)) ((0 0) (2 0) (0 2)))
 (((0 0) (2 0) (2 2)) ((0 0) (2 0) (1 1)))
 (((0 0) (2 0) (2 2)) ((0 0) (2 2) (0 2)))
 (((0 0) (2 0) (2 2)) ((0 0) (0 2) (1 1)))
 (((0 0) (2 0) (2 2)) ((2 0) (2 2) (0 2)))
 (((0 0) (2 0) (2 2)) ((2 0) (2 2) (1 1)))
 (((0 0) (2 0) (2 2)) ((2 2) (0 2) (1 1)))
 (((0 0) (2 0) (0 2)) ((0 0) (2 0) (1 1)))
 (((0 0) (2 0) (0 2)) ((0 0) (2 2) (0 2)))
 (((0 0) (2 0) (0 2)) ((0 0) (0 2) (1 1)))
 (((0 0) (2 0) (0 2)) ((2 0) (2 2) (0 2)))
 (((0 0) (2 0) (0 2)) ((2 0) (2 2) (1 1)))
 (((0 0) (2 0) (0 2)) ((2 2) (0 2) (1 1)))
 (((0 0) (2 0) (1 1)) ((0 0) (2 2) (0 2)))
 (((0 0) (2 0) (1 1)) ((0 0) (0 2) (1 1)))
 (((0 0) (2 0) (1 1)) ((2 0) (2 2) (0 2)))
 (((0 0) (2 0) (1 1)) ((2 0) (2 2) (1 1)))
 (((0 0) (2 0) (1 1)) ((2 2) (0 2) (1 1)))
 (((0 0) (2 2) (0 2)) ((0 0) (0 2) (1 1)))
 (((0 0) (2 2) (0 2)) ((2 0) (2 2) (0 2)))
 (((0 0) (2 2) (0 2)) ((2 0) (2 2) (1 1)))
 (((0 0) (2 2) (0 2)) ((2 2) (0 2) (1 1)))
 (((0 0) (0 2) (1 1)) ((2 0) (2 2) (0 2)))
 (((0 0) (0 2) (1 1)) ((2 0) (2 2) (1 1)))
 (((0 0) (0 2) (1 1)) ((2 2) (0 2) (1 1)))
 (((2 0) (2 2) (0 2)) ((2 0) (2 2) (1 1)))
 (((2 0) (2 2) (0 2)) ((2 2) (0 2) (1 1)))
 (((2 0) (2 2) (1 1)) ((2 2) (0 2) (1 1))))
"=== Тест 14: Много точек с дубликатами ==="
"Цель: Проверить обработку дублирующихся точек"
"Пункт C"
"Треугольники:"
(((0 0) (3 0) (0 4)) ((0 0) (3 0) (1 1)) ((0 0) (0 4) (1 1))
 ((3 0) (0 4) (1 1)))
"Равнобедренные:"
NIL
"Равносторонние:"
NIL
"Прямоугольные:"
(((0 0) (3 0) (0 4)))
```

```
"Подобные:"
NIL
"=== Тест 15: Комплексный тест со всеми типами ==="
"Цель: Комплексная проверка всех функций анализа треугольников"
"Пункт C"
"Треугольники:"
(((0 0) (4 0) (0 3)) ((0 0) (4 0) (6 1.732)) ((0 0) (4 0) (10 4))
 ((0 0) (4 0) (0 5)) ((0 0) (4 0) (3 5)) ((0 0) (4 0) (0 9))
 ((0 0) (4 0) (1 1)) ((0 0) (4 0) (2 2)) ((0 0) (4 0) (3 3))
 ((0 0) (0 3) (5 0)) ((0 0) (0 3) (7 0)) ((0 0) (0 3) (6 1.732))
 ((0 0) (0 3) (8 0)) ((0 0) (0 3) (12 0)) ((0 0) (0 3) (10 4))
 ((0 0) (0 3) (3 5)) ((0 0) (0 3) (1 1)) ((0 0) (0 3) (2 2))
 ((0 0) (0 3) (3 3)) ((0 0) (5 0) (6 1.732)) ((0 0) (5 0) (10 4))
 ((0 0) (5 0) (0 5)) ((0 0) (5 0) (3 5)) ((0 0) (5 0) (0 9))
 ((0 0) (5 0) (1 1)) ((0 0) (5 0) (2 2)) ((0 0) (5 0) (3 3))
 ((0 0) (7 0) (6 1.732)) ((0 0) (7 0) (10 4)) ((0 0) (7 0) (0 5))
 ((0 0) (7 0) (3 5)) ((0 0) (7 0) (0 9)) ((0 0) (7 0) (1 1))
 ((0 0) (7 0) (2 2)) ((0 0) (7 0) (3 3)) ((0 0) (6 1.732) (8 0))
 ((0 0) (6 1.732) (12 0)) ((0 0) (6 1.732) (10 4)) ((0 0) (6
1.732) (0 5))
 ((0 0) (6 1.732) (3 5)) ((0 0) (6 1.732) (0 9)) ((0 0) (6 1.732)
(1 1))
 ((0 0) (6 1.732) (2 2)) ((0 0) (6 1.732) (3 3)) ((0 0) (8 0) (10
4))
 ((0 0) (8 0) (0 5)) ((0 0) (8 0) (3 5)) ((0 0) (8 0) (0 9))
 ((0 0) (8 0) (1 1)) ((0 0) (8 0) (2 2)) ((0 0) (8 0) (3 3))
 ((0 0) (12 0) (10 4)) ((0 0) (12 0) (0 5)) ((0 0) (12 0) (3 5))
 ((0 0) (12 0) (0 9)) ((0 0) (12 0) (1 1)) ((0 0) (12 0) (2 2))
 ((0 0) (12 0) (3 3)) ((0 0) (10 4) (0 5)) ((0 0) (10 4) (3 5))
 ((0 0) (10 4) (0 9)) ((0 0) (10 4) (1 1)) ((0 0) (10 4) (2 2))
 ((0 0) (10 4) (3 3)) ((0 0) (0 5) (3 5)) ((0 0) (0 5) (1 1))
 ((0 0) (0 5) (2 2)) ((0 0) (0 5) (3 3)) ((0 0) (3 5) (0 9))
 ((0 0) (3 5) (1 1)) ((0 0) (3 5) (2 2)) ((0 0) (3 5) (3 3))
 ((0 0) (0 9) (1 1)) ((0 0) (0 9) (2 2)) ((0 0) (0 9) (3 3))
 ((4 0) (0 3) (5 0)) ((4 0) (0 3) (7 0)) ((4 0) (0 3) (6 1.732))
 ((4 0) (0 3) (8 0)) ((4 0) (0 3) (12 0)) ((4 0) (0 3) (10 4))
 ((4 0) (0 3) (0 5)) ((4 0) (0 3) (3 5)) ((4 0) (0 3) (0 9))
 ((4 0) (0 3) (1 1)) ((4 0) (0 3) (2 2)) ((4 0) (0 3) (3 3))
 ((4 0) (5 0) (6 1.732)) ((4 0) (5 0) (10 4)) ((4 0) (5 0) (0 5))
 ((4 0) (5 0) (3 5)) ((4 0) (5 0) (0 9)) ((4 0) (5 0) (1 1))
 ((4 0) (5 0) (2 2)) ((4 0) (5 0) (3 3)) ((4 0) (7 0) (6 1.732))
 ((4 0) (7 0) (10 4)) ((4 0) (7 0) (0 5)) ((4 0) (7 0) (3 5))
 ((4 0) (7 0) (0 9)) ((4 0) (7 0) (1 1)) ((4 0) (7 0) (2 2))
 ((4 0) (7 0) (3 3)) ((4 0) (6 1.732) (8 0)) ((4 0) (6 1.732) (12
0))
 ((4 0) (6 1.732) (10 4)) ((4 0) (6 1.732) (0 5)) ((4 0) (6 1.732)
(3 5))
 ((4 0) (6 1.732) (0 9)) ((4 0) (6 1.732) (1 1)) ((4 0) (6 1.732)
(2 2))
 ((4 0) (6 1.732) (3 3)) ((4 0) (8 0) (10 4)) ((4 0) (8 0) (0 5))
 ((4 0) (8 0) (3 5)) ((4 0) (8 0) (0 9)) ((4 0) (8 0) (1 1))
```

((4 0) (8 0) (2 2)) ((4 0) (8 0) (3 3)) ((4 0) (12 0) (10 4))
((4 0) (12 0) (0 5)) ((4 0) (12 0) (3 5)) ((4 0) (12 0) (0 9))
((4 0) (12 0) (1 1)) ((4 0) (12 0) (2 2)) ((4 0) (12 0) (3 3))
((4 0) (10 4) (0 5)) ((4 0) (10 4) (3 5)) ((4 0) (10 4) (0 9))
((4 0) (10 4) (1 1)) ((4 0) (10 4) (2 2)) ((4 0) (10 4) (3 3))
((4 0) (0 5) (3 5)) ((4 0) (0 5) (0 9)) ((4 0) (0 5) (1 1))
((4 0) (0 5) (2 2)) ((4 0) (0 5) (3 3)) ((4 0) (3 5) (0 9))
((4 0) (3 5) (1 1)) ((4 0) (3 5) (2 2)) ((4 0) (3 5) (3 3))
((4 0) (0 9) (1 1)) ((4 0) (0 9) (2 2)) ((4 0) (0 9) (3 3))
((4 0) (1 1) (2 2)) ((4 0) (1 1) (3 3)) ((4 0) (2 2) (3 3))
((0 3) (5 0) (7 0)) ((0 3) (5 0) (6 1.732)) ((0 3) (5 0) (8 0))
((0 3) (5 0) (12 0)) ((0 3) (5 0) (10 4)) ((0 3) (5 0) (0 5))
((0 3) (5 0) (3 5)) ((0 3) (5 0) (0 9)) ((0 3) (5 0) (1 1))
((0 3) (5 0) (2 2)) ((0 3) (5 0) (3 3)) ((0 3) (7 0) (6 1.732))
((0 3) (7 0) (8 0)) ((0 3) (7 0) (12 0)) ((0 3) (7 0) (10 4))
((0 3) (7 0) (0 5)) ((0 3) (7 0) (3 5)) ((0 3) (7 0) (0 9))
((0 3) (7 0) (1 1)) ((0 3) (7 0) (2 2)) ((0 3) (7 0) (3 3))
((0 3) (6 1.732) (8 0)) ((0 3) (6 1.732) (12 0)) ((0 3) (6 1.732)
(10 4))
((0 3) (6 1.732) (0 5)) ((0 3) (6 1.732) (3 5)) ((0 3) (6 1.732)
(0 9))
((0 3) (6 1.732) (1 1)) ((0 3) (6 1.732) (2 2)) ((0 3) (6 1.732)
(3 3))
((0 3) (8 0) (12 0)) ((0 3) (8 0) (10 4)) ((0 3) (8 0) (0 5))
((0 3) (8 0) (3 5)) ((0 3) (8 0) (0 9)) ((0 3) (8 0) (1 1))
((0 3) (8 0) (2 2)) ((0 3) (8 0) (3 3)) ((0 3) (12 0) (10 4))
((0 3) (12 0) (0 5)) ((0 3) (12 0) (3 5)) ((0 3) (12 0) (0 9))
((0 3) (12 0) (1 1)) ((0 3) (12 0) (2 2)) ((0 3) (12 0) (3 3))
((0 3) (10 4) (0 5)) ((0 3) (10 4) (3 5)) ((0 3) (10 4) (0 9))
((0 3) (10 4) (1 1)) ((0 3) (10 4) (2 2)) ((0 3) (10 4) (3 3))
((0 3) (0 5) (3 5)) ((0 3) (0 5) (1 1)) ((0 3) (0 5) (2 2))
((0 3) (0 5) (3 3)) ((0 3) (3 5) (0 9)) ((0 3) (3 5) (1 1))
((0 3) (3 5) (2 2)) ((0 3) (3 5) (3 3)) ((0 3) (0 9) (1 1))
((0 3) (0 9) (2 2)) ((0 3) (0 9) (3 3)) ((0 3) (1 1) (2 2))
((0 3) (1 1) (3 3)) ((0 3) (2 2) (3 3)) ((5 0) (7 0) (6 1.732))
((5 0) (7 0) (10 4)) ((5 0) (7 0) (0 5)) ((5 0) (7 0) (3 5))
((5 0) (7 0) (0 9)) ((5 0) (7 0) (1 1)) ((5 0) (7 0) (2 2))
((5 0) (7 0) (3 3)) ((5 0) (6 1.732) (8 0)) ((5 0) (6 1.732) (12
0))
((5 0) (6 1.732) (10 4)) ((5 0) (6 1.732) (0 5)) ((5 0) (6 1.732)
(3 5))
((5 0) (6 1.732) (0 9)) ((5 0) (6 1.732) (1 1)) ((5 0) (6 1.732)
(2 2))
((5 0) (6 1.732) (3 3)) ((5 0) (8 0) (10 4)) ((5 0) (8 0) (0 5))
((5 0) (8 0) (3 5)) ((5 0) (8 0) (0 9)) ((5 0) (8 0) (1 1))
((5 0) (8 0) (2 2)) ((5 0) (8 0) (3 3)) ((5 0) (12 0) (10 4))
((5 0) (12 0) (0 5)) ((5 0) (12 0) (3 5)) ((5 0) (12 0) (0 9))
((5 0) (12 0) (1 1)) ((5 0) (12 0) (2 2)) ((5 0) (12 0) (3 3))
((5 0) (10 4) (0 5)) ((5 0) (10 4) (3 5)) ((5 0) (10 4) (0 9))
((5 0) (10 4) (1 1)) ((5 0) (10 4) (2 2)) ((5 0) (10 4) (3 3))
((5 0) (0 5) (3 5)) ((5 0) (0 5) (0 9)) ((5 0) (0 5) (1 1))

((5 0) (0 5) (2 2)) ((5 0) (0 5) (3 3)) ((5 0) (3 5) (0 9))
((5 0) (3 5) (1 1)) ((5 0) (3 5) (2 2)) ((5 0) (3 5) (3 3))
((5 0) (0 9) (1 1)) ((5 0) (0 9) (2 2)) ((5 0) (0 9) (3 3))
((5 0) (1 1) (2 2)) ((5 0) (1 1) (3 3)) ((5 0) (2 2) (3 3))
((7 0) (6 1.732) (8 0)) ((7 0) (6 1.732) (12 0)) ((7 0) (6 1.732)
(10 4))
((7 0) (6 1.732) (0 5)) ((7 0) (6 1.732) (3 5)) ((7 0) (6 1.732)
(0 9))
((7 0) (6 1.732) (1 1)) ((7 0) (6 1.732) (2 2)) ((7 0) (6 1.732)
(3 3))
((7 0) (8 0) (10 4)) ((7 0) (8 0) (0 5)) ((7 0) (8 0) (3 5))
((7 0) (8 0) (0 9)) ((7 0) (8 0) (1 1)) ((7 0) (8 0) (2 2))
((7 0) (8 0) (3 3)) ((7 0) (12 0) (10 4)) ((7 0) (12 0) (0 5))
((7 0) (12 0) (3 5)) ((7 0) (12 0) (0 9)) ((7 0) (12 0) (1 1))
((7 0) (12 0) (2 2)) ((7 0) (12 0) (3 3)) ((7 0) (10 4) (0 5))
((7 0) (10 4) (3 5)) ((7 0) (10 4) (0 9)) ((7 0) (10 4) (1 1))
((7 0) (10 4) (2 2)) ((7 0) (10 4) (3 3)) ((7 0) (0 5) (3 5))
((7 0) (0 5) (0 9)) ((7 0) (0 5) (1 1)) ((7 0) (0 5) (2 2))
((7 0) (0 5) (3 3)) ((7 0) (3 5) (0 9)) ((7 0) (3 5) (1 1))
((7 0) (3 5) (2 2)) ((7 0) (3 5) (3 3)) ((7 0) (0 9) (1 1))
((7 0) (0 9) (2 2)) ((7 0) (0 9) (3 3)) ((7 0) (1 1) (2 2))
((7 0) (1 1) (3 3)) ((7 0) (2 2) (3 3)) ((6 1.732) (8 0) (12 0))
((6 1.732) (8 0) (10 4)) ((6 1.732) (8 0) (0 5)) ((6 1.732) (8 0)
(3 5))
((6 1.732) (8 0) (0 9)) ((6 1.732) (8 0) (1 1)) ((6 1.732) (8 0)
(2 2))
((6 1.732) (8 0) (3 3)) ((6 1.732) (12 0) (10 4)) ((6 1.732) (12
0) (0 5))
((6 1.732) (12 0) (3 5)) ((6 1.732) (12 0) (0 9)) ((6 1.732) (12
0) (1 1))
((6 1.732) (12 0) (2 2)) ((6 1.732) (12 0) (3 3)) ((6 1.732) (10
4) (0 5))
((6 1.732) (10 4) (3 5)) ((6 1.732) (10 4) (0 9)) ((6 1.732) (10
4) (1 1))
((6 1.732) (10 4) (2 2)) ((6 1.732) (10 4) (3 3)) ((6 1.732) (0
5) (3 5))
((6 1.732) (0 5) (0 9)) ((6 1.732) (0 5) (1 1)) ((6 1.732) (0 5)
(2 2))
((6 1.732) (0 5) (3 3)) ((6 1.732) (3 5) (0 9)) ((6 1.732) (3 5)
(1 1))
((6 1.732) (3 5) (2 2)) ((6 1.732) (3 5) (3 3)) ((6 1.732) (0 9)
(1 1))
((6 1.732) (0 9) (2 2)) ((6 1.732) (0 9) (3 3)) ((6 1.732) (1 1)
(2 2))
((6 1.732) (1 1) (3 3)) ((6 1.732) (2 2) (3 3)) ((8 0) (12 0) (10
4))
((8 0) (12 0) (0 5)) ((8 0) (12 0) (3 5)) ((8 0) (12 0) (0 9))
((8 0) (12 0) (1 1)) ((8 0) (12 0) (2 2)) ((8 0) (12 0) (3 3))
((8 0) (10 4) (0 5)) ((8 0) (10 4) (3 5)) ((8 0) (10 4) (0 9))
((8 0) (10 4) (1 1)) ((8 0) (10 4) (2 2)) ((8 0) (10 4) (3 3))
((8 0) (0 5) (3 5)) ((8 0) (0 5) (0 9)) ((8 0) (0 5) (1 1))

```
 ((8 0) (0 5) (2 2)) ((8 0) (0 5) (3 3)) ((8 0) (3 5) (0 9))
 ((8 0) (3 5) (1 1)) ((8 0) (3 5) (2 2)) ((8 0) (3 5) (3 3))
 ((8 0) (0 9) (1 1)) ((8 0) (0 9) (2 2)) ((8 0) (0 9) (3 3))
 ((8 0) (1 1) (2 2)) ((8 0) (1 1) (3 3)) ((8 0) (2 2) (3 3))
 ((12 0) (10 4) (0 5)) ((12 0) (10 4) (3 5)) ((12 0) (10 4) (0 9))
 ((12 0) (10 4) (1 1)) ((12 0) (10 4) (2 2)) ((12 0) (10 4) (3 3))
 ((12 0) (0 5) (3 5)) ((12 0) (0 5) (0 9)) ((12 0) (0 5) (1 1))
 ((12 0) (0 5) (2 2)) ((12 0) (0 5) (3 3)) ((12 0) (3 5) (0 9))
 ((12 0) (3 5) (1 1)) ((12 0) (3 5) (2 2)) ((12 0) (3 5) (3 3))
 ((12 0) (0 9) (1 1)) ((12 0) (0 9) (2 2)) ((12 0) (0 9) (3 3))
 ((12 0) (1 1) (2 2)) ((12 0) (1 1) (3 3)) ((12 0) (2 2) (3 3))
 ((10 4) (0 5) (3 5)) ((10 4) (0 5) (0 9)) ((10 4) (0 5) (1 1))
 ((10 4) (0 5) (2 2)) ((10 4) (0 5) (3 3)) ((10 4) (3 5) (0 9))
 ((10 4) (3 5) (1 1)) ((10 4) (3 5) (2 2)) ((10 4) (3 5) (3 3))
 ((10 4) (0 9) (1 1)) ((10 4) (0 9) (2 2)) ((10 4) (0 9) (3 3))
 ((10 4) (1 1) (2 2)) ((10 4) (1 1) (3 3)) ((10 4) (2 2) (3 3))
 ((0 5) (3 5) (0 9)) ((0 5) (3 5) (1 1)) ((0 5) (3 5) (2 2))
 ((0 5) (3 5) (3 3)) ((0 5) (0 9) (1 1)) ((0 5) (0 9) (2 2))
 ((0 5) (0 9) (3 3)) ((0 5) (1 1) (2 2)) ((0 5) (1 1) (3 3))
 ((0 5) (2 2) (3 3)) ((3 5) (0 9) (1 1)) ((3 5) (0 9) (2 2))
 ((3 5) (0 9) (3 3)) ((3 5) (1 1) (2 2)) ((3 5) (1 1) (3 3))
 ((3 5) (2 2) (3 3)) ((0 9) (1 1) (2 2)) ((0 9) (1 1) (3 3))
 ((0 9) (2 2) (3 3)))
"Равнобедренные:"
(((0 0) (4 0) (2 2)) ((0 0) (0 3) (3 3)) ((0 0) (5 0) (0 5))
 ((0 0) (6 1.732) (12 0)) ((4 0) (6 1.732) (8 0)) ((4 0) (1 1) (3
3))
 ((0 3) (10 4) (0 5)) ((0 3) (1 1) (2 2)) ((5 0) (7 0) (6 1.732))
 ((5 0) (0 5) (1 1)) ((5 0) (0 5) (2 2)) ((5 0) (0 5) (3 3))
 ((5 0) (2 2) (3 3)) ((7 0) (12 0) (10 4)) ((7 0) (12 0) (3 3))
 ((7 0) (10 4) (3 3)) ((8 0) (12 0) (10 4)) ((8 0) (10 4) (3 5))
 ((8 0) (3 5) (1 1)) ((10 4) (3 5) (3 3)) ((0 5) (2 2) (3 3)))
"Равносторонние:"
(((5 0) (7 0) (6 1.732)))
"Прямоугольные:"
(((0 0) (4 0) (0 3)) ((0 0) (4 0) (0 5)) ((0 0) (4 0) (0 9))
 ((0 0) (4 0) (2 2)) ((0 0) (0 3) (5 0)) ((0 0) (0 3) (7 0))
 ((0 0) (0 3) (8 0)) ((0 0) (0 3) (12 0)) ((0 0) (0 3) (3 3))
 ((0 0) (5 0) (0 5)) ((0 0) (5 0) (0 9)) ((0 0) (7 0) (0 5))
 ((0 0) (7 0) (0 9)) ((0 0) (8 0) (0 5)) ((0 0) (8 0) (0 9))
 ((0 0) (12 0) (0 5)) ((0 0) (12 0) (0 9)) ((0 0) (0 5) (3 5))
 ((4 0) (1 1) (2 2)) ((4 0) (2 2) (3 3)) ((0 3) (0 5) (3 5))
 ((0 3) (0 5) (3 3)) ((0 3) (3 5) (3 3)) ((0 3) (0 9) (3 3))
 ((7 0) (10 4) (3 3)) ((8 0) (10 4) (0 9)) ((8 0) (3 5) (2 2))
 ((0 5) (3 5) (0 9)) ((0 5) (3 5) (3 3)))
"Подобные:"
((((0 0) (4 0) (0 3)) ((0 0) (12 0) (0 9)))
 (((0 0) (4 0) (0 3)) ((0 5) (3 5) (0 9)))
 (((0 0) (4 0) (6 1.732)) ((6 1.732) (8 0) (12 0)))
 (((0 0) (4 0) (10 4)) ((5 0) (7 0) (2 2)))
 (((0 0) (4 0) (10 4)) ((5 0) (3 5) (3 3)))
```

```
(((0 0) (4 0) (3 5)) ((5 0) (1 1) (3 3)))
(((0 0) (4 0) (3 5)) ((0 5) (1 1) (3 3)))
(((0 0) (4 0) (1 1)) ((0 0) (8 0) (2 2)))
(((0 0) (4 0) (1 1)) ((0 0) (12 0) (3 3)))
(((0 0) (4 0) (2 2)) ((0 0) (0 3) (3 3)))
(((0 0) (4 0) (2 2)) ((0 0) (5 0) (0 5)))
(((0 0) (4 0) (2 2)) ((7 0) (10 4) (3 3)))
(((0 0) (4 0) (3 3)) ((0 0) (0 3) (2 2)))
(((0 0) (4 0) (3 3)) ((0 3) (1 1) (3 3)))
(((0 0) (0 3) (5 0)) ((0 0) (0 5) (3 5)))
(((0 0) (0 3) (3 5)) ((0 3) (5 0) (3 3)))
(((0 0) (0 3) (3 5)) ((5 0) (8 0) (3 3)))
(((0 0) (0 3) (1 1)) ((0 0) (0 9) (3 3)))
(((0 0) (0 3) (1 1)) ((0 3) (2 2) (3 3)))
(((0 0) (0 3) (1 1)) ((8 0) (10 4) (1 1)))
(((0 0) (0 3) (1 1)) ((12 0) (10 4) (3 3)))
(((0 0) (0 3) (1 1)) ((10 4) (3 5) (1 1)))
(((0 0) (0 3) (2 2)) ((0 3) (1 1) (3 3)))
(((0 0) (0 3) (3 3)) ((0 0) (5 0) (0 5)))
(((0 0) (0 3) (3 3)) ((7 0) (10 4) (3 3)))
(((0 0) (5 0) (6 1.732)) ((7 0) (6 1.732) (12 0)))
(((0 0) (5 0) (0 5)) ((7 0) (10 4) (3 3)))
(((0 0) (5 0) (1 1)) ((0 0) (0 5) (1 1)))
(((0 0) (5 0) (1 1)) ((0 3) (5 0) (0 5)))
(((0 0) (5 0) (1 1)) ((8 0) (3 5) (3 3)))
(((0 0) (5 0) (2 2)) ((0 0) (0 5) (2 2)))
(((0 0) (5 0) (2 2)) ((4 0) (8 0) (3 5)))
(((0 0) (5 0) (3 3)) ((0 0) (0 5) (3 3)))
(((0 0) (5 0) (3 3)) ((10 4) (0 9) (1 1)))
(((0 0) (7 0) (6 1.732)) ((5 0) (6 1.732) (12 0)))
(((0 0) (6 1.732) (8 0)) ((4 0) (6 1.732) (12 0)))
(((0 0) (8 0) (2 2)) ((0 0) (12 0) (3 3)))
(((0 0) (12 0) (0 9)) ((0 5) (3 5) (0 9)))
(((0 0) (10 4) (2 2)) ((0 3) (5 0) (7 0)))
(((0 0) (10 4) (3 3)) ((4 0) (0 3) (7 0)))
(((0 0) (10 4) (3 3)) ((0 3) (7 0) (3 3)))
(((0 0) (0 5) (1 1)) ((0 3) (5 0) (0 5)))
(((0 0) (0 5) (1 1)) ((8 0) (3 5) (3 3)))
(((0 0) (0 5) (2 2)) ((4 0) (8 0) (3 5)))
(((0 0) (0 5) (3 3)) ((10 4) (0 9) (1 1)))
(((0 0) (3 5) (1 1)) ((4 0) (5 0) (1 1)))
(((0 0) (3 5) (1 1)) ((0 3) (12 0) (3 3)))
(((0 0) (3 5) (2 2)) ((0 3) (0 5) (1 1)))
(((0 0) (0 9) (3 3)) ((0 3) (2 2) (3 3)))
(((0 0) (0 9) (3 3)) ((8 0) (10 4) (1 1)))
(((0 0) (0 9) (3 3)) ((12 0) (10 4) (3 3)))
(((0 0) (0 9) (3 3)) ((10 4) (3 5) (1 1)))
(((4 0) (0 3) (5 0)) ((7 0) (8 0) (3 3)))
(((4 0) (0 3) (5 0)) ((10 4) (2 2) (3 3)))
(((4 0) (0 3) (7 0)) ((0 3) (7 0) (3 3)))
(((4 0) (0 3) (0 5)) ((5 0) (7 0) (10 4)))
```

```
(((4 0) (0 3) (0 5)) ((7 0) (3 5) (3 3)))
(((4 0) (0 3) (3 5)) ((4 0) (10 4) (3 5)))
(((4 0) (0 3) (1 1)) ((4 0) (8 0) (2 2)))
(((4 0) (0 3) (1 1)) ((12 0) (0 9) (3 3)))
(((4 0) (0 3) (1 1)) ((10 4) (3 5) (0 9)))
(((4 0) (0 3) (1 1)) ((3 5) (1 1) (3 3)))
(((4 0) (0 3) (1 1)) ((3 5) (2 2) (3 3)))
(((4 0) (0 3) (2 2)) ((4 0) (8 0) (1 1)))
(((4 0) (0 3) (3 3)) ((4 0) (7 0) (3 3)))
(((4 0) (0 3) (3 3)) ((10 4) (0 9) (3 3)))
(((4 0) (5 0) (6 1.732)) ((7 0) (6 1.732) (8 0)))
(((4 0) (5 0) (0 5)) ((7 0) (8 0) (3 5)))
(((4 0) (5 0) (1 1)) ((0 3) (12 0) (3 3)))
(((4 0) (5 0) (2 2)) ((4 0) (12 0) (2 2)))
(((4 0) (5 0) (2 2)) ((12 0) (0 5) (2 2)))
(((4 0) (7 0) (6 1.732)) ((5 0) (6 1.732) (8 0)))
(((4 0) (7 0) (0 5)) ((7 0) (0 5) (3 5)))
(((4 0) (7 0) (3 5)) ((4 0) (0 5) (3 5)))
(((4 0) (7 0) (3 3)) ((10 4) (0 9) (3 3)))
(((4 0) (8 0) (10 4)) ((4 0) (3 5) (2 2)))
(((4 0) (8 0) (10 4)) ((0 3) (0 5) (2 2)))
(((4 0) (8 0) (10 4)) ((3 5) (0 9) (1 1)))
(((4 0) (8 0) (2 2)) ((12 0) (0 9) (3 3)))
(((4 0) (8 0) (2 2)) ((10 4) (3 5) (0 9)))
(((4 0) (8 0) (2 2)) ((3 5) (1 1) (3 3)))
(((4 0) (8 0) (2 2)) ((3 5) (2 2) (3 3)))
(((4 0) (12 0) (2 2)) ((12 0) (0 5) (2 2)))
(((4 0) (10 4) (1 1)) ((7 0) (12 0) (0 9)))
(((4 0) (10 4) (2 2)) ((5 0) (1 1) (2 2)))
(((4 0) (10 4) (2 2)) ((0 5) (1 1) (2 2)))
(((4 0) (10 4) (3 3)) ((7 0) (10 4) (0 9)))
(((4 0) (3 5) (1 1)) ((0 3) (3 5) (2 2)))
(((4 0) (3 5) (2 2)) ((0 3) (0 5) (2 2)))
(((4 0) (3 5) (2 2)) ((3 5) (0 9) (1 1)))
(((4 0) (3 5) (3 3)) ((8 0) (12 0) (2 2)))
(((4 0) (1 1) (2 2)) ((4 0) (2 2) (3 3)))
(((4 0) (1 1) (2 2)) ((0 3) (0 9) (3 3)))
(((4 0) (1 1) (2 2)) ((8 0) (3 5) (2 2)))
(((4 0) (1 1) (3 3)) ((7 0) (12 0) (10 4)))
(((4 0) (1 1) (3 3)) ((8 0) (12 0) (10 4)))
(((4 0) (2 2) (3 3)) ((0 3) (0 9) (3 3)))
(((4 0) (2 2) (3 3)) ((8 0) (3 5) (2 2)))
(((0 3) (5 0) (8 0)) ((0 3) (8 0) (3 3)))
(((0 3) (5 0) (0 5)) ((8 0) (3 5) (3 3)))
(((0 3) (5 0) (3 3)) ((5 0) (8 0) (3 3)))
(((0 3) (7 0) (0 5)) ((7 0) (1 1) (2 2)))
(((0 3) (12 0) (0 9)) ((8 0) (1 1) (3 3)))
(((0 3) (10 4) (3 5)) ((10 4) (0 5) (3 3)))
(((0 3) (10 4) (3 3)) ((10 4) (0 5) (3 5)))
(((0 3) (0 5) (3 5)) ((0 3) (0 5) (3 3)))
(((0 3) (0 5) (3 5)) ((0 3) (3 5) (3 3)))
```

```
(((0 3) (0 5) (3 5)) ((0 5) (3 5) (3 3)))
(((0 3) (0 5) (2 2)) ((3 5) (0 9) (1 1)))
(((0 3) (0 5) (3 3)) ((0 3) (3 5) (3 3)))
(((0 3) (0 5) (3 3)) ((0 5) (3 5) (3 3)))
(((0 3) (3 5) (3 3)) ((0 5) (3 5) (3 3)))
(((0 3) (0 9) (3 3)) ((8 0) (3 5) (2 2)))
(((0 3) (1 1) (2 2)) ((8 0) (10 4) (3 5)))
(((0 3) (1 1) (2 2)) ((8 0) (3 5) (1 1)))
(((0 3) (2 2) (3 3)) ((8 0) (10 4) (1 1)))
(((0 3) (2 2) (3 3)) ((12 0) (10 4) (3 3)))
(((0 3) (2 2) (3 3)) ((10 4) (3 5) (1 1)))
(((5 0) (7 0) (10 4)) ((7 0) (3 5) (3 3)))
(((5 0) (7 0) (0 9)) ((0 9) (1 1) (2 2)))
(((5 0) (7 0) (2 2)) ((5 0) (3 5) (3 3)))
(((5 0) (8 0) (0 5)) ((8 0) (0 5) (3 5)))
(((5 0) (8 0) (3 5)) ((5 0) (0 5) (3 5)))
(((5 0) (0 5) (0 9)) ((8 0) (12 0) (3 5)))
(((5 0) (0 5) (2 2)) ((5 0) (0 5) (3 3)))
(((5 0) (3 5) (2 2)) ((8 0) (0 9) (1 1)))
(((5 0) (1 1) (2 2)) ((0 5) (1 1) (2 2)))
(((5 0) (1 1) (3 3)) ((0 5) (1 1) (3 3)))
(((5 0) (2 2) (3 3)) ((0 5) (2 2) (3 3)))
(((7 0) (8 0) (10 4)) ((8 0) (1 1) (2 2)))
(((7 0) (8 0) (3 3)) ((10 4) (2 2) (3 3)))
(((7 0) (12 0) (10 4)) ((8 0) (12 0) (10 4)))
(((8 0) (10 4) (3 5)) ((8 0) (3 5) (1 1)))
(((8 0) (10 4) (1 1)) ((12 0) (10 4) (3 3)))
(((8 0) (10 4) (1 1)) ((10 4) (3 5) (1 1)))
(((12 0) (10 4) (3 5)) ((3 5) (0 9) (2 2)))
(((12 0) (10 4) (0 9)) ((10 4) (1 1) (3 3)))
(((12 0) (10 4) (0 9)) ((3 5) (0 9) (3 3)))
(((12 0) (10 4) (3 3)) ((10 4) (3 5) (1 1)))
(((12 0) (3 5) (3 3)) ((0 9) (2 2) (3 3)))
(((12 0) (0 9) (3 3)) ((10 4) (3 5) (0 9)))
(((12 0) (0 9) (3 3)) ((3 5) (1 1) (3 3)))
(((12 0) (0 9) (3 3)) ((3 5) (2 2) (3 3)))
(((10 4) (3 5) (0 9)) ((3 5) (1 1) (3 3)))
(((10 4) (3 5) (0 9)) ((3 5) (2 2) (3 3)))
(((10 4) (1 1) (3 3)) ((3 5) (0 9) (3 3)))
(((3 5) (1 1) (3 3)) ((3 5) (2 2) (3 3))))
```

Вывод: Все ответы верны

6.3 Пограмма для пункта С

(defun build_triangle(l) (build_triangles (free_points (rem_dup l))))

(defun rem_dup (l)(cond
  ((null l) nil)
  ((find_dup (car l) (cdr l)) (rem_dup (cdr l)))

```
   (T (cons (car l) (rem_dup (cdr l))))
))

(defun find_dup (e l)(cond
   ((null l) nil)
   ((equal e (car l)) T)
   (T (find_dup e (cdr l)))
))

(defun build_triangles (l)(cond
   ((null l) nil)
   (T (append (check_triangles (car l)) (build_triangles (cdr l))))
))

(defun check_triangles (tri)(cond
   ((null tri) nil)
   ((= (area tri) 0) nil)
   (T (list tri))
))

(defun area (tri)(abs(/ (+ (* (caar tri) (- (cadadr tri) (cadr(caddr tri))))
   (* (caadr tri) (- (cadr(caddr tri)) (cadar tri)))
   (* (caaddr tri) (- (cadar tri) (cadadr tri)))) 2)))

(defun free_points (l)(cond
   ((null (cddr l)) nil)
   (T (append (free_dop (car l) (cdr l)) (free_points (cdr l))))
))

(defun free_dop (a l)(cond
   ((null (cdr l)) nil)
   (T (append (free_ddop a (car l) (cdr l)) (free_dop a (cdr l))))
))

(defun free_ddop (a b l)(cond
   ((null l) nil)
   (T (cons (list a b (car l)) (free_ddop a b (cdr l))))
))

(defun distance (p1 p2)
  (sqrt (+ (* (- (car p2) (car p1)) (- (car p2) (car p1)))
           (* (- (cadr p2) (cadr p1)) (- (cadr p2) (cadr p1))))))

(defun approx_equal (a b) (< (abs (- a b)) 0.001))

(defun equilateral (l)(cond
   ((null l) nil)
   ((equilateral1 (car l)) (cons (car l) (equilateral (cdr l))))
```

```
   (T (equilateral (cdr l)))
))

(defun equilateral1 (l)(cond
   ((and (approx_equal (distance (car l) (cadr l)) (distance (car l) (caddr l)))
         (approx_equal (distance (car l) (cadr l)) (distance (cadr l) (caddr l)))) T)
))

(defun isosceles (l)(cond
   ((null l) nil)
   ((isosceles1 (car l)) (cons (car l) (isosceles (cdr l))))
   (T (isosceles (cdr l)))
))

(defun isosceles1 (l)(cond
   ((or (approx_equal (distance (car l) (cadr l)) (distance (car l) (caddr l)))
        (approx_equal (distance (car l) (cadr l)) (distance (cadr l) (caddr l)))
        (approx_equal (distance (cadr l) (caddr l)) (distance (car l) (caddr l)))) T)
))

(defun right_triangles (l)(cond
   ((null l) nil)
   ((right_triangle_p (car l)) (cons (car l) (right_triangles (cdr l))))
   (T (right_triangles (cdr l)))
))

(defun right_triangle_p (tri)
  (right_triangle_p1 (distance (car tri) (cadr tri))
                     (distance (cadr tri) (caddr tri))
                     (distance (caddr tri) (car tri))))

(defun right_triangle_p1 (a b c)
  (or (approx_equal (+ (* a a) (* b b)) (* c c))
      (approx_equal (+ (* a a) (* c c)) (* b b))
      (approx_equal (+ (* b b) (* c c)) (* a a))))

(defun similar_triangles (l)(cond
   ((null l) nil)
   (T (append (find_similar (car l) (cdr l)) (similar_triangles (cdr l))))
))

(defun find_similar (t1 l)(cond
   ((null l) nil)
   ((similar_p t1 (car l)) (cons (list t1 (car l)) (find_similar t1 (cdr l))))
   (T (find_similar t1 (cdr l)))
))

(defun similar_p (t1 t2)
```

```
  (similar_check (sort3 (distance (car t1) (cadr t1)) (distance (cadr t1) (caddr t1)) (distance (caddr t1) (car t1)))
                 (sort3 (distance (car t2) (cadr t2)) (distance (cadr t2) (caddr t2)) (distance (caddr t2) (car t2)))))

(defun similar_check (s1 s2)(cond
    ((<= (car s2) 0) nil)
    ((and (approx_equal (/ (car s1) (car s2)) (/ (cadr s1) (cadr s2)))
          (approx_equal (/ (car s1) (car s2)) (/ (caddr s1) (caddr s2)))) T)
))

(defun sort3 (a b c)(cond
    ((and (<= a b) (<= b c)) (list a b c))
    ((and (<= a c) (<= c b)) (list a c b))
    ((and (<= b a) (<= a c)) (list b a c))
    ((and (<= b c) (<= c a)) (list b c a))
    ((and (<= c a) (<= a b)) (list c a b))
    (T (list c b a))
))

(defun triangleC (l)
  (print "Пункт C")
  (print "Треугольники:") (print (build_triangle l))
  (print "Равнобедренные:") (print (isosceles (build_triangle l)))
  (print "Равносторонние:") (print (equilateral (build_triangle l)))
  (print "Прямоугольные:") (print (right_triangles (build_triangle l)))
  (print "Подобные:") (print (similar_triangles (build_triangle l)))
)
;; Тест 1: Нормальные треугольники - проверка основного функционала
(print "=== Тест 1: Нормальные треугольники ===")
(print "Цель: Проверить корректное построение треугольников и их анализ")
(triangleC '((0 0) (3 0) (0 4) (1 1) (2 2)))

;; Тест 2: Равносторонний треугольник
(print "=== Тест 2: Равносторонний треугольник ===")
(print "Цель: Проверить обнаружение равносторонних треугольников")
(triangleC '((0 0) (2 0) (1 1.732) (1 0) (0 1)))

;; Тест 3: Равнобедренные треугольники
(print "=== Тест 3: Равнобедренные треугольники ===")
(print "Цель: Проверить обнаружение равнобедренных треугольников")
(triangleC '((0 0) (4 0) (2 3) (1 0) (0 2)))

;; Тест 4: Прямоугольные треугольники
(print "=== Тест 4: Прямоугольные треугольники ===")
(print "Цель: Проверить обнаружение прямоугольных треугольников")
(triangleC '((0 0) (3 0) (0 4) (1 0) (0 1)))
```

```
;; Тест 5: Подобные треугольники
(print "=== Тест 5: Подобные треугольники ===")
(print "Цель: Проверить обнаружение подобных треугольников")
(triangleC '((0 0) (3 0) (0 4)
             (0 0) (6 0) (0 8)
             (1 1) (4 1) (1 5)))

;; Тест 6: Точки на одной прямой
(print "=== Тест 6: Точки на одной прямой ===")
(print "Цель: Проверить обработку вырожденных случаев")
(triangleC '((0 0) (1 1) (2 2) (3 3) (4 4)))

;; Тест 7: Все точки совпадают
(print "=== Тест 7: Все точки совпадают ===")
(print "Цель: Проверить обработку идентичных точек")
(triangleC '((0 0) (0 0) (0 0) (0 0) (0 0)))

;; Тест 8: Только 3 точки
(print "=== Тест 8: Только 3 точки ===")
(print "Цель: Проверить работу с минимальным количеством точек")
(triangleC '((0 0) (3 0) (0 4)))

;; Тест 9: Большие координаты
(print "=== Тест 9: Большие координаты ===")
(print "Цель: Проверить устойчивость к большим значениям")
(triangleC '((1000 1000) (3000 1000) (1000 4000) (2000 2000) (1500 1500)))

;; Тест 10: Отрицательные координаты
(print "=== Тест 10: Отрицательные координаты ===")
(print "Цель: Проверить работу с отрицательными координатами")
(triangleC '((-2 -2) (1 -2) (-2 1) (0 0) (-1 -1)))

;; Тест 11: Разные типы треугольников в одном наборе
(print "=== Тест 11: Разные типы треугольников в одном наборе ===")
(print "Цель: Проверить классификацию смешанного набора треугольников")
(triangleC '((0 0) (3 0) (0 4)          ; прямоугольный
             (0 0) (2 0) (1 1.732)      ; равносторонний
             (0 0) (4 0) (2 3)          ; равнобедренный
             (1 1) (2 2) (3 3)))        ; вырожденный

;; Тест 12: Точки образуют только вырожденные треугольники
(print "=== Тест 12: Точки образуют только вырожденные треугольники ===")
(print "Цель: Проверить обработку когда все треугольники вырожденные")
(triangleC '((0 0) (1 1) (2 2) (3 3) (0 1) (1 2)))

;; Тест 13: Точки в вершинах квадрата
(print "=== Тест 13: Точки в вершинах квадрата ===")
(print "Цель: Проверить построение треугольников из точек квадрата")
```

```
(triangleC '((0 0) (2 0) (2 2) (0 2) (1 1)))

;; Тест 14: Много точек с дубликатами
(print "=== Тест 14: Много точек с дубликатами ===")
(print "Цель: Проверить обработку дублирующихся точек")
(triangleC '((0 0) (3 0) (0 4) (0 0) (3 0) (0 4) (1 1) (1 1)))

;; Тест 15: Комплексный тест со всеми типами
(print "=== Тест 15: Комплексный тест со всеми типами ===")
(print "Цель: Комплексная проверка всех функций анализа треугольников")
(triangleC '((0 0) (4 0) (0 3)          ; прямоугольный
             (5 0) (7 0) (6 1.732)      ; равносторонний
             (8 0) (12 0) (10 4)        ; равнобедренный
             (0 5) (3 5) (0 9)          ; прямоугольный (подобный первому)
             (1 1) (2 2) (3 3)))        ; вырожденный
```

7.Итог пункт E

7.1 Тесты для пункта E

```
;; Тест 1: Нормальные окружности - внешнее расположение
(print "=== Тест 1: Нормальные окружности - внешнее расположение ===")
(print "Цель: Проверить построение касательной окружности для внешне
расположенных окружностей")
(punE '((0 0) (1 1) (0 2)) '((5 0) (6 1) (5 2)))

;; Тест 2: Окружности разного размера
(print "=== Тест 2: Окружности разного размера ===")
(print "Цель: Проверить построение для окружностей с разными радиусами")
(punE '((0 0) (1 0) (0 1)) '((4 0) (6 0) (4 3)))

;; Тест 3: Окружности пересекаются
(print "=== Тест 3: Окружности пересекаются ===")
(print "Цель: Проверить обработку пересекающихся окружностей")
(punE '((0 0) (2 0) (0 2)) '((1 1) (3 1) (1 3)))

;; Тест 4: Одна окружность внутри другой
(print "=== Тест 4: Одна окружность внутри другой ===")
(print "Цель: Проверить обработку вложенных окружностей")
(punE '((0 0) (3 0) (0 3)) '((1 1) (1.5 1) (1 1.5)))

;; Тест 5: Концентрические окружности
(print "=== Тест 5: Концентрические окружности ===")
(print "Цель: Проверить обработку концентрических окружностей")
(punE '((0 0) (1 0) (0 1)) '((0 0) (2 0) (0 2)))
```

```
;; Тест 6: Касающиеся окружности
(print "=== Тест 6: Касающиеся окружности ===")
(print "Цель: Проверить обработку касающихся окружностей")
(punE '((0 0) (2 0) (0 2)) '((4 0) (6 0) (4 2)))

;; Тест 7: Большие окружности
(print "=== Тест 7: Большие окружности ===")
(print "Цель: Проверить устойчивость к большим значениям")
(punE '((100 100) (200 100) (100 200)) '((500 100) (600 100) (500 200)))

;; Тест 8: Отрицательные координаты
(print "=== Тест 8: Отрицательные координаты ===")
(print "Цель: Проверить работу с отрицательными координатами")
(punE '((-2 -2) (-1 -1) (-2 0)) '((2 2) (3 3) (2 4)))

;; Тест 9: Окружности на большом расстоянии
(print "=== Тест 9: Окружности на большом расстоянии ===")
(print "Цель: Проверить построение для далеко расположенных окружностей")
(punE '((0 0) (1 0) (0 1)) '((10 0) (11 0) (10 1)))

;; Тест 10: Почти касающиеся окружности
(print "=== Тест 10: Почти касающиеся окружности ===")
(print "Цель: Проверить граничный случай близких окружностей")
(punE '((0 0) (2 0) (0 2)) '((5 0) (7 0) (5 2.1)))

;; Тест 11: Только одна окружность строится
(print "=== Тест 11: Только одна окружность строится ===")
(print "Цель: Проверить обработку частично успешного построения")
(punE '((0 0) (1 1) (2 2)) '((0 0) (2 0) (0 2)))

;; Тест 12: Обе окружности не строятся
(print "=== Тест 12: Обе окружности не строятся ===")
(print "Цель: Проверить обработку полной неудачи построения")
(punE '((0 0) (1 1) (2 2)) '((3 3) (4 4) (5 5)))

;; Тест 13: Окружности с центрами на оси X
(print "=== Тест 13: Окружности с центрами на оси X ===")
(print "Цель: Проверить специальное расположение центров")
(punE '((-2 0) (-1 0) (0 0)) '((3 0) (4 0) (5 0)))

;; Тест 14: Окружности с центрами на оси Y
(print "=== Тест 14: Окружности с центрами на оси Y ===")
(print "Цель: Проверить вертикальное расположение центров")
(punE '((0 -2) (0 -1) (0 0)) '((0 3) (0 4) (0 5)))
```

```
;; Тест 15: Комплексный тест со всеми случаями
(print "=== Тест 15: Комплексный тест со всеми случаями ===")
(print "Цель: Комплексная проверка всех функций анализа окружностей")
(punE '((0 0) (2 0) (0 2)) '((5 0) (7 0) (5 2)))
```

7.2 Результаты тестов

```
"=== Тест 1: Нормальные окружности - внешнее расположение ==="
"Цель: Проверить построение касательной окружности для внешне
расположенных окружностей"
"Пункт E"
"Окружность 1"
((0 1) 1)
"Окружность 2"
((5 1) 1)
"Отношение окружностей"
"Окружности не пересекаются"
"Анализ центров"
(("Центр первой:" (0 1)) ("Центр второй:" (5 1))
 "Ни одна окружность не находится внутри другой")
"Касательная окружность"
"=== Тест 2: Окружности разного размера ==="
"Цель: Проверить построение для окружностей с разными радиусами"
"Пункт E"
"Окружность 1"
((1/2 1/2) 0.70710677)
"Окружность 2"
((5 3/2) 1.8027756)
"Отношение окружностей"
"Окружности не пересекаются"
"Анализ центров"
(("Центр первой:" (1/2 1/2)) ("Центр второй:" (5 3/2))
 "Ни одна окружность не находится внутри другой")
"Касательная окружность"
"=== Тест 3: Окружности пересекаются ==="
"Цель: Проверить обработку пересекающихся окружностей"
"Пункт E"
"Окружность 1"
((1 1) 1.4142135)
"Окружность 2"
((2 2) 1.4142135)
"Отношение окружностей"
("Окружности пересекаются в двух точках"
 ((3.3660254 1.6339746) (1.6339746 3.3660254)))
"Анализ центров"
(("Центр первой:" (1 1)) ("Центр второй:" (2 2))
 "Ни одна окружность не находится внутри другой")
"Касательная окружность"
"=== Тест 4: Одна окружность внутри другой ==="
"Цель: Проверить обработку вложенных окружностей"
```

```
"Пункт E"
"Окружность 1"
((3/2 3/2) 2.1213202)
"Окружность 2"
((1.25 1.25) 0.35355338)
"Отношение окружностей"
"Одна окружность внутри другой"
"Анализ центров"
(("Центр первой:" (3/2 3/2)) ("Центр второй:" (1.25 1.25))
 "Вторая окружность находится внутри первой")
"Касательная окружность"
"=== Тест 5: Концентрические окружности ==="
"Цель: Проверить обработку концентрических окружностей"
"Пункт E"
"Окружность 1"
((1/2 1/2) 0.70710677)
"Окружность 2"
((1 1) 1.4142135)
"Отношение окружностей"
("Окружности касаются" (3.0 3.0))
"Анализ центров"
(("Центр первой:" (1/2 1/2)) ("Центр второй:" (1 1))
 "Ни одна окружность не находится внутри другой")
"Касательная окружность"
"=== Тест 6: Касающиеся окружности ==="
"Цель: Проверить обработку касающихся окружностей"
"Пункт E"
"Окружность 1"
((1 1) 1.4142135)
"Окружность 2"
((5 1) 1.4142135)
"Отношение окружностей"
"Окружности не пересекаются"
"Анализ центров"
(("Центр первой:" (1 1)) ("Центр второй:" (5 1))
 "Ни одна окружность не находится внутри другой")
"Касательная окружность"
"=== Тест 7: Большие окружности ==="
"Цель: Проверить устойчивость к большим значениям"
"Пункт E"
"Окружность 1"
((150 150) 70.71068)
"Окружность 2"
((550 150) 70.71068)
"Отношение окружностей"
"Окружности не пересекаются"
"Анализ центров"
(("Центр первой:" (150 150)) ("Центр второй:" (550 150))
 "Ни одна окружность не находится внутри другой")
"Касательная окружность"
"=== Тест 8: Отрицательные координаты ==="
```

```
"Цель: Проверить работу с отрицательными координатами"
"Пункт E"
"Окружность 1"
((-2 -1) 1)
"Окружность 2"
((2 3) 1)
"Отношение окружностей"
"Окружности не пересекаются"
"Анализ центров"
(("Центр первой:" (-2 -1)) ("Центр второй:" (2 3))
 "Ни одна окружность не находится внутри другой")
"Касательная окружность"
"=== Тест 9: Окружности на большом расстоянии ==="
"Цель: Проверить построение для далеко расположенных окружностей"
"Пункт E"
"Окружность 1"
((1/2 1/2) 0.70710677)
"Окружность 2"
((21/2 1/2) 0.70710677)
"Отношение окружностей"
"Окружности не пересекаются"
"Анализ центров"
(("Центр первой:" (1/2 1/2)) ("Центр второй:" (21/2 1/2))
 "Ни одна окружность не находится внутри другой")
"Касательная окружность"
"=== Тест 10: Почти касающиеся окружности ==="
"Цель: Проверить граничный случай близких окружностей"
"Пункт E"
"Окружность 1"
((1 1) 1.4142135)
"Окружность 2"
((6.0 1.05) 1.4499999)
"Отношение окружностей"
"Окружности не пересекаются"
"Анализ центров"
(("Центр первой:" (1 1)) ("Центр второй:" (6.0 1.05))
 "Ни одна окружность не находится внутри другой")
"Касательная окружность"
"=== Тест 11: Только одна окружность строится ==="
"Цель: Проверить обработку частично успешного построения"
"Пункт E"
"Окружность 1"
"Невозможно построить окружность!"
"Окружность 2"
((1 1) 1.4142135)
"Окружность 2 построена"
((1 1) 1.4142135)
"=== Тест 12: Обе окружности не строятся ==="
"Цель: Проверить обработку полной неудачи построения"
"Пункт E"
"Окружность 1"
```

```
"Невозможно построить окружность!"
"Окружность 2"
"Невозможно построить окружность!"
"=== Тест 13: Окружности с центрами на оси X ==="
"Цель: Проверить специальное расположение центров"
"Пункт E"
"Окружность 1"
"Невозможно построить окружность!"
"Окружность 2"
"Невозможно построить окружность!"
"=== Тест 14: Окружности с центрами на оси Y ==="
"Цель: Проверить вертикальное расположение центров"
"Пункт E"
"Окружность 1"
"Невозможно построить окружность!"
"Окружность 2"
"Невозможно построить окружность!"
"=== Тест 15: Комплексный тест со всеми случаями ==="
"Цель: Комплексная проверка всех функций анализа окружностей"
"Пункт E"
"Окружность 1"
((1 1) 1.4142135)
"Окружность 2"
((6 1) 1.4142135)
"Отношение окружностей"
"Окружности не пересекаются"
"Анализ центров"
(("Центр первой:" (1 1)) ("Центр второй:" (6 1))
 "Ни одна окружность не находится внутри другой")
"Касательная окружность"
```

Вывод: Все тесты работают верно

7.3 Код пункта E

```
(defun punE (l1 l2)
        (print "Пункт E")
        (print "Окружность 1")
        (print (circle_build l1))
        (print "Окружность 2")
        (print (circle_build l2))
        (edop (circle_build l1)(circle_build l2))
)

(defun edop (circle1 circle2)(cond
   ((and (listp circle1) (listp circle2))
           (print "Отношение окружностей")
      (print (circles_relation circle1 circle2))
      (print "Анализ центров")
      (print (analyze circle1 circle2))
```

```
      (print "Касательная окружность")
      (tangent_check circle1 circle2)
         )
         ((listp circle1)
            (print "Окружность 1 построена")
      (print circle1)
         )
         ((listp circle2)
      (print "Окружность 2 построена")
      (print circle2)
         )
         (T "Ошибка построения окружностей!")
))

(defun circle_build (l)(cond
   ((not (can_build (car l) (cadr l) (caddr l))) "Невозможно построить окружность!")
   (T (list
        (list
         (formula_for_circle (caar l) (caadr l) (caaddr l) (cadar l) (cadadr l) (car (cdaddr l)))
         (formula_for_circle (cadar l) (cadadr l) (car (cdaddr l)) (caar l) (caadr l) (caaddr l))
        )
        (radius (formula_for_circle (caar l) (caadr l) (caaddr l) (cadar l) (cadadr l) (car (cdaddr l)))
              (formula_for_circle (cadar l) (cadadr l) (car (cdaddr l)) (caar l) (caadr l) (caaddr l))
              (car l)
        )
     ))
))

(defun formula_for_circle (a b c d e f)
   (/
      (- (* (- e d) (+ (- (* f f) (* d d)) (- (* c c) (* a a))))
        (* (- f d) (+ (- (* e e) (* d d)) (- (* b b) (* a a)))))
      (* 2 (- (* (- c a) (- e d))
              (* (- b a) (- f d))))
   )
)

(defun radius (x y point)
   (sqrt (+ (* (- x (car point)) (- x (car point)))
         (* (- y (cadr point)) (- y (cadr point)))))
)

(defun distance (p1 p2)
   (sqrt (+ (* (- (car p2) (car p1)) (- (car p2) (car p1)))
         (* (- (cadr p2) (cadr p1)) (- (cadr p2) (cadr p1)))))
)

(defun calc_a (x1 y1 r1 x2 y2 r2)
```

```
   (/ (+ (* r1 r1) (- (* (distance (list x1 y1) (list x2 y2)) (distance (list x1 y1) (list x2 y2)))) (* r2 r2))
     (* 2 (distance (list x1 y1) (list x2 y2))))
)

(defun calc_h (x1 y1 r1 x2 y2 r2)(cond
   ((< (- (* r1 r1) (* (calc_a x1 y1 r1 x2 y2 r2) (calc_a x1 y1 r1 x2 y2 r2))) 0) 0)
   (T (sqrt (- (* r1 r1) (* (calc_a x1 y1 r1 x2 y2 r2) (calc_a x1 y1 r1 x2 y2 r2)))))
))

(defun find_points (x1 y1 r1 x2 y2 r2)
   (list
      (list
         (+ (+ x1 (/ (calc_a x1 y1 r1 x2 y2 r2) (distance (list x1 y1) (list x2 y2))) (- x2 x1))
           (/ (* (calc_h x1 y1 r1 x2 y2 r2) (- y2 y1)) (distance (list x1 y1) (list x2 y2))))
         (- (+ y1 (/ (calc_a x1 y1 r1 x2 y2 r2) (distance (list x1 y1) (list x2 y2))) (- y2 y1))
           (/ (* (calc_h x1 y1 r1 x2 y2 r2) (- x2 x1)) (distance (list x1 y1) (list x2 y2)))))
      (list
         (- (+ x1 (/ (calc_a x1 y1 r1 x2 y2 r2) (distance (list x1 y1) (list x2 y2))) (- x2 x1))
           (/ (* (calc_h x1 y1 r1 x2 y2 r2) (- y2 y1)) (distance (list x1 y1) (list x2 y2))))
         (+ (+ y1 (/ (calc_a x1 y1 r1 x2 y2 r2) (distance (list x1 y1) (list x2 y2))) (- y2 y1))
           (/ (* (calc_h x1 y1 r1 x2 y2 r2) (- x2 x1)) (distance (list x1 y1) (list x2 y2))))))
)

(defun circles_relation (circle1 circle2)(cond
   ((and (= (distance (car circle1) (car circle2)) 0) (= (cadr circle1) (cadr circle2))) "Окружности
совпадают")
   ((> (distance (car circle1) (car circle2)) (+ (cadr circle1) (cadr circle2))) "Окружности не
пересекаются")
   ((< (distance (car circle1) (car circle2)) (abs (- (cadr circle1) (cadr circle2)))) "Одна окружность
внутри другой")
   ((or (= (distance (car circle1) (car circle2)) (+ (cadr circle1) (cadr circle2)))
       (= (distance (car circle1) (car circle2)) (abs (- (cadr circle1) (cadr circle2)))))
       (list "Окружности касаются" (car (find_points (caar circle1) (cadar circle1) (cadr circle1)
                    (caar circle2) (cadar circle2) (cadr circle2)))))
   (T (list "Окружности пересекаются в двух точках" (find_points (caar circle1) (cadar circle1)
(cadr circle1)
                                        (caar circle2) (cadar circle2) (cadr circle2))))
))

(defun can_build (p1 p2 p3)
   (not (eq 0 (- (* (- (car p2) (car p1)) (- (cadr p3) (cadr p1)))
            (* (- (cadr p2) (cadr p1)) (- (car p3) (car p1)))))))
)

(defun inside (c1 c2)(cond
   ((>= (distance (car c1) (car c2)) (abs (- (cadr c1) (cadr c2)))) "Ни одна окружность не
находится внутри другой")
   ((< (cadr c1) (cadr c2)) "Первая окружность находится внутри второй")
```

```
   (T "Вторая окружность находится внутри первой")
))

(defun analyze (c1 c2)
   (list
      (list "Центр первой:" (car c1))
      (list "Центр второй:" (car c2))
      (inside c1 c2)
   )
)

(defun tangent_center (circle1 circle2)
   (list
      (+ (caar circle1)
        (/ (* (/ (- (distance (car circle1) (car circle2))
                    (cadr circle1)
                    (cadr circle2))
                 2)
              (- (caar circle2) (caar circle1)))
           (distance (car circle1) (car circle2))))
      (+ (cadar circle1)
        (/ (* (/ (- (distance (car circle1) (car circle2))
                    (cadr circle1)
                    (cadr circle2))
                 2)
              (- (cadar circle2) (cadar circle1)))
           (distance (car circle1) (car circle2))))
   )
)

(defun tangent_radius (circle1 circle2)
   (/ (- (distance (car circle1) (car circle2))
        (cadr circle1)
        (cadr circle2))
      2)
)

(defun tangent_circle (circle1 circle2)
   (list
      (list (tangent_center circle1 circle2) (tangent_radius circle1 circle2))
      (list "Точка касания с первой:" (car (find_points (caar circle1) (cadar circle1) (cadr circle1)
                                          (caar circle2) (cadar circle2) (cadr circle2))))
      (list "Точка касания со второй:" (car (find_points (caar circle2) (cadar circle2) (cadr circle2)
                                          (caar circle1) (cadar circle1) (cadr circle1))))
   )
)

(defun tangent_check (circle1 circle2)(cond
```

```
    ((and (equal (circles_relation circle1 circle2) "Окружности не пересекаются")
          (equal (inside circle1 circle2) "Ни одна окружность не находится внутри другой"))
        (tangent_circle circle1 circle2))
    (T "Нельзя построить касательную окружность")
))

;; Тест 1: Нормальные окружности - внешнее расположение
(print "=== Тест 1: Нормальные окружности - внешнее расположение ===")
(print "Цель: Проверить построение касательной окружности для внешне расположенных окружностей")
(punE '((0 0) (1 1) (0 2)) '((5 0) (6 1) (5 2)))

;; Тест 2: Окружности разного размера
(print "=== Тест 2: Окружности разного размера ===")
(print "Цель: Проверить построение для окружностей с разными радиусами")
(punE '((0 0) (1 0) (0 1)) '((4 0) (6 0) (4 3)))

;; Тест 3: Окружности пересекаются
(print "=== Тест 3: Окружности пересекаются ===")
(print "Цель: Проверить обработку пересекающихся окружностей")
(punE '((0 0) (2 0) (0 2)) '((1 1) (3 1) (1 3)))

;; Тест 4: Одна окружность внутри другой
(print "=== Тест 4: Одна окружность внутри другой ===")
(print "Цель: Проверить обработку вложенных окружностей")
(punE '((0 0) (3 0) (0 3)) '((1 1) (1.5 1) (1 1.5)))

;; Тест 5: Концентрические окружности
(print "=== Тест 5: Концентрические окружности ===")
(print "Цель: Проверить обработку концентрических окружностей")
(punE '((0 0) (1 0) (0 1)) '((0 0) (2 0) (0 2)))

;; Тест 6: Касающиеся окружности
(print "=== Тест 6: Касающиеся окружности ===")
(print "Цель: Проверить обработку касающихся окружностей")
(punE '((0 0) (2 0) (0 2)) '((4 0) (6 0) (4 2)))

;; Тест 7: Большие окружности
(print "=== Тест 7: Большие окружности ===")
(print "Цель: Проверить устойчивость к большим значениям")
(punE '((100 100) (200 100) (100 200)) '((500 100) (600 100) (500 200)))

;; Тест 8: Отрицательные координаты
(print "=== Тест 8: Отрицательные координаты ===")
(print "Цель: Проверить работу с отрицательными координатами")
(punE '((-2 -2) (-1 -1) (-2 0)) '((2 2) (3 3) (2 4)))

;; Тест 9: Окружности на большом расстоянии
```

```
(print "=== Тест 9: Окружности на большом расстоянии ===")
(print "Цель: Проверить построение для далеко расположенных окружностей")
(punE '((0 0) (1 0) (0 1)) '((10 0) (11 0) (10 1)))

;; Тест 10: Почти касающиеся окружности
(print "=== Тест 10: Почти касающиеся окружности ===")
(print "Цель: Проверить граничный случай близких окружностей")
(punE '((0 0) (2 0) (0 2)) '((5 0) (7 0) (5 2.1)))

;; Тест 11: Только одна окружность строится
(print "=== Тест 11: Только одна окружность строится ===")
(print "Цель: Проверить обработку частично успешного построения")
(punE '((0 0) (1 1) (2 2)) '((0 0) (2 0) (0 2)))

;; Тест 12: Обе окружности не строятся
(print "=== Тест 12: Обе окружности не строятся ===")
(print "Цель: Проверить обработку полной неудачи построения")
(punE '((0 0) (1 1) (2 2)) '((3 3) (4 4) (5 5)))

;; Тест 13: Окружности с центрами на оси X
(print "=== Тест 13: Окружности с центрами на оси X ===")
(print "Цель: Проверить специальное расположение центров")
(punE '((-2 0) (-1 0) (0 0)) '((3 0) (4 0) (5 0)))

;; Тест 14: Окружности с центрами на оси Y
(print "=== Тест 14: Окружности с центрами на оси Y ===")
(print "Цель: Проверить вертикальное расположение центров")
(punE '((0 -2) (0 -1) (0 0)) '((0 3) (0 4) (0 5)))

;; Тест 15: Комплексный тест со всеми случаями
(print "=== Тест 15: Комплексный тест со всеми случаями ===")
(print "Цель: Комплексная проверка всех функций анализа окружностей")
(punE '((0 0) (2 0) (0 2)) '((5 0) (7 0) (5 2)))
```