МИНИСТЕРСТВО ЦИФРОВОГО РАЗВИТИЯ, СВЯЗИ И МАССОВЫХ КОММУНИКАЦИЙ РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ТЕЛЕКОММУНИКАЦИЙ ИМ. ПРОФ. М.А. БОНЧ-БРУЕВИЧА»
(СПбГУТ)
ФАКУЛЬТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ И ПРОГРАММНОЙ ИНЖЕНЕРИИ **(ИТПИ)**
Кафедра программной инженерии и вычислительной техники **(ПИиВТ)**

.

Дисциплина: «Разработка приложений искусственного интеллекта в киберфизических системах»
Лабораторная работа №2.

Тема: «Определение координат точек пересечения окружности со сторонами ромба»

Выполнил:

Студент группы ИКПИ-23

Харлова А.А

Подпись _____

Принял:

Ерофеев С.А.

Подпись _____

2024 г.

Постановка задачи

Написать программу на языке PROLOG, которая определяет точки пересечения окружности со сторонами ромба. Окружность задаётся координатами цента и радиусом, ромб - четырьмя вершинами.

Алгоритм решения

Задача определения координат точек пересечения окружности со сторонами ромба подразумевает аналитическое решение геометрической задачи. В ходе выполнения работы необходимо решить систему уравнений, содержащую уравнение окружности и прямых, на которых лежат заданные отрезки (стороны ромба), предварительно вывев их по известным координатам. Алгоритм решения кратко представлен на рисунке 1.

Описание программы

В таблице 1 приведено описание термов, используемых в программе.

Терм	Тип	Описание
realList	real*	Список чисел типа "real". Хранит координаты точки на плоскости в формате "[X, Y]", где X - положение точки по оси абсцисс, Y- положение точки по оси ординат соответственно.
realList2D	realList*	Двумерный список чисел типа "real". Хранит координаты точек на плоскости в формате списка: "[[X_1, Y_1], Point2, …]", где "[[X_1, Y_1]" или "Point2" - данные типа realList.
stream	file	Стандартный предикат для обработки файлов, который позволяет использовать файлы произвольного доступа.

Таблица 1. Используемые термы

Программа разделена на 4 модуля: "IINTERSECTION.PRO", "MATH.PRO", "STRUCTS.PRO", "IO.PRO". В таблицах 2-5 приведено описание предикатов, используемых в каждом модуле программы.

Модуль "IINTERSECTION.PRO" является основным и содержит термы, указанные в таблице 1, точку входа в программу, реализованный алгоритм для решения задачи, предикаты для работы с геометрическими формулами и утверждениями. Описание предикатов данного модуля – таблица 2.

Предикат	Описание
start()	Запуск программы
getLenth(real, real, real)	Длина одномерного отрезка
getLenth2D(realList, realList, real)	Длина двумерного отрезка
sumPoints(realList, realList, realList)	Сложение векторов на плоскости

isRhombus(realList2D)	Ромб
isCircle(real)	Окружность
lineEquation(real, real, real, real)	Уравнение прямой с известными коэффициентами
lineEquationFromTwoPoints(realList, realList, real, real,real)	Уравнение прямой, проходящей через две точки
onSegment(realList, realList, realList)	Принадлежность точки отрезку
getOnSegmentPoint(realList, realList, realList, realList2D)	Предикат, конвертирующий координаты точки в формат "realList2D", если она принадлежнит отрезку
getIntersections(realList2D, realList, real, realList2D)	Пересение окружности со сторонами ромба
getSegmentIntersection(realList, realList, realList, real, realList2D)	Перечение окружности с отрезком
getLineIntersection(realList, real, real, real, real, realList, realList)	Пересечение окружности с прямой

Таблица 2. Предикаты модуля "IINTERSECTION.PRO"

В модуль "MATH.PRO" включены предикаты для работы с основными математическими операциями и алгебраическими функциями. Описание предикатов данного модуля – таблица 3.

Предикат	Описание
normalErrorRate(real, real)	Погрешность вычислений
equal(real, real)	Равнозначимость двух элементов
equal(real, real, real, real)	Равнозначимость четырех элементов
equalOrHigher(real, real)	Оператор >=
max(real, real, real)	Максимум двух значений
min(real, real, real)	Минимум двух значений
qEquationRoot(real, real, real, real)	Формула корня квадратного уравнения
qEquationTwoRoots(real, real, real, real, real)	Формула нахождения корней квадратного уравнения в случае, когда два корня различны
qEquationOneRoot(real, real, real, real)	Формула нахождения корней квадратного уравнения в случае, когда два корня идентичны
qEquationSolver(real, real, real, real, real, real)	Решение квадратного уравнения
qEquation(real, real, real, real, real)	Квадратное уравнение

Таблица 3. Предикаты модуля "MATH.PRO"

Модуль "STRUCTS.PRO" определяет предикаты структур данных, такие как множество, и взаимодействие с ними. Описание предикатов данного модуля – таблица 4.

Предикат	Описание
member(realList, realList2D)	Элемент списка или множества
set(realList2D, realList2D)	Конвертер списка в множество
emptyList(realList)	Пустой список
append(realList2D, realList2D, realList2D)	Добавление в список

Таблица 4. Предикаты модуля "STRUCTS.PRO"

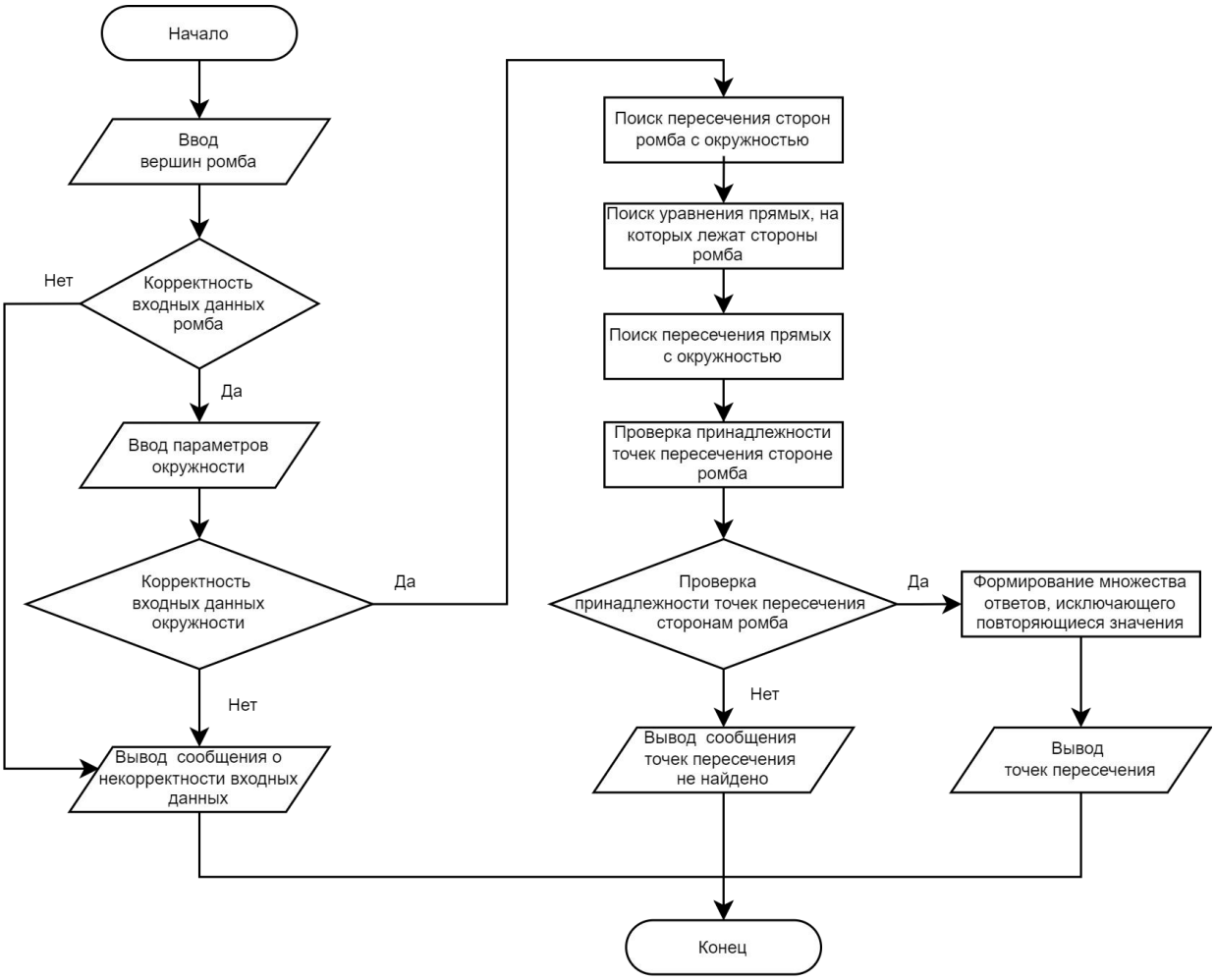
В четвертом модуле "IO.PRO" реализованы предикаты для взаимодействия программы с пользователем, например, ввод и вывод данных, отображение меню программы. Описание предикатов данного модуля – таблица 5.

Предикат	Описание
menu(integer)	Пользовательское меню
inputData(realList2D, realList, real, integer)	Входные данные
inputFilePath(string, string)	Путь к файлу
inputRhombus(realList2D)	Ввод координат вершин ромба
inputPoint(realList, string)	Ввод координат точки
inputCircle(realList, real)	Ввод параметров окружности
correctMenuOption(integer)	Корректность выбора опции пользовательского меню
correctInput(realList2D, real, realList2D)	Корректность введенных данных
correctRhombus(realList2D, realList2D)	Корректность введенных координат вершин ромба
correctCircle(real)	Корректность введенных параметров окружности
outputData(realList2D, realList, real, integer)	Выходные данные
outputPoints(realList2D)	Форматированный вывод координат точки
outputAnswer(realList2D, integer)	Форматированный вывод результата работы программы
outputRhombus(reallist2D)	Форматированный вывод координат вершин ромба

outputCircle(realList, real)	Форматированный вывод параметров окружности

Таблица 5. Предикаты модуля "IO.PRO"

Алгоритм решения



Используемые формулы

В ходе разработки программы были использованы математические формулы, на основе которых было выведено уравнение для решения задачи. Все применённые формулы представлены в таблице 6.

Формула	Описание
$gy + kx + b = 0$	Общий вид уравнения прямой
$y = kx + b$	Уравнение наклонной прямой
$y = \frac{y_2 - y_1}{x_2 - x_1} \cdot x + \frac{x_2 y_1 - x_1 y_2}{x_2 - x_1}, x_2 \neq x_1$	Уравнение прямой, заданное двумя точками
$r^2 = (x - x_0)^2 + (y - y_0)^2$	Уравнение окружности

$Ax^2 + Bx + C = 0$	Формула квадратного уравнения
$D = B^2 - 4AC$	Формула нахождения дискриминанта
$x_{1,2} = \frac{-B \pm \sqrt{D}}{2A},\ D >= 0$	Формула нахождения корней квадратного уравнения через дискриминант
$((\frac{k}{g})^2 + 1) \cdot x^2 + (2k \cdot \frac{b + y_0 * g}{g^2} - 2x_0) \cdot x + + (\frac{b}{g} + y_0)^2 + x_0{}^2 - r^2 = 0,\ g \neq 0$	Выведенная формула для определения координат пересечения окружности с горизонтальной или наклонной прямой
$y^2 - 2y_0 \cdot y + y_0{}^2 + (b - x_0)^2 - r^2 = 0$	Выведенная формула для определения координат пересечения окружности с вертикальной прямой

Таблица 6. Используемые формулы

Вывод

Была разработана программа на языке PROLOG, определяющая точки пересечения окружности со сторонами ромба. В ходе разработки были введены необходимые предикаты. Выполнено тестирование. Результаты тестирования приведены в отчёте. Поставленная задача выполнена.

Код программы

Модуль "IINTERSECTION.PRO":

```
DOMAINS
 realList = real*
 realList2D = realList*
 file = stream

CONSTANTS
% Точность вычислений
 currency = 0.00001

PREDICATES
 start()

 getLenth(real, real, real)
 getLenth2D(realList, realList, real)
 sumPoints(realList, realList, realList)

 isRhombus(realList2D)
 isCircle(real)

 lineEquation(real, real, real, real)
 lineEquationFromTwoPoints(realList, realList, real, real,real)
 onSegment(realList, realList, realList)
 getOnSegmentPoint(realList, realList, realList, realList2D)
```

```
 getIntersections(realList2D, realList, real, realList2D)
 getSegmentIntersection(realList, realList, realList, real, realList2D)
 getLineIntersection(realList, real, real, real, real, realList, realList)

INCLUDE "STRUCTS.PRO"
INCLUDE "MATH.PRO"
INCLUDE "IO.PRO"

GOAL
 start.

CLAUSES

% ---------------------- Начало программы ----------------------

 start() :-
        menu(Option),
        inputData(Rhombus, CircleCentre, Radius, Option),
        outputData(Rhombus, CircleCentre, Radius, 1),
        getIntersections(Rhombus, CircleCentre, Radius, Answer),
        outputAnswer(Answer, 1),
        outputAnswer(Answer, Option);
        write("Try again..."), nl.

% --------------------- Геометрические операции ----------------------

% Определение длины одномерного отрезка
 getLenth(X1, X2, Lenth):-
        Lenth = abs(X1 - X2).

% Определение длины двумерного отрезка
 getLenth2D([X1, Y1], [X2, Y2], Lenth):-
        getLenth(X1, X2, XLenth),
        getLenth(Y1, Y2, YLenth),
        Lenth = sqrt(XLenth * XLenth + YLenth * YLenth).

% Сложение координат двух точек
 sumPoints([Point1X, Point1Y], [Point2X, Point2Y], [PointX, PointY]):-
        PointX = Point1X + Point2X,
        PointY = Point1Y + Point2Y.

% Условие существования ромба
 isRhombus([Point1, Point2, Point3, Point4]):-
        % Все стороны равны
        getLenth2D(Point1, Point2, Lenth12),
        getLenth2D(Point2, Point3, Lenth23),
        getLenth2D(Point3, Point4, Lenth34),
        getLenth2D(Point4, Point1, Lenth41),

        equal(Lenth12, Lenth23, Lenth34, Lenth41),

        % Диагонали пересекаются в их центрах
```

```
        sumPoints(Point1, Point3, LeftPoint),
        sumPoints(Point2, Point4, RightPoint),

        LeftPoint = RightPoint.

% Условие существования окружности
 isCircle(Radius):-
        Radius > 0.

% Уравнение наклонной прямой
 lineEquation(K, B, X, Y):-
        bound(X), Y = K * X + B.

% Уравнения прямой, заданное двумя известными точками
% kX + gY + b = 0
 lineEquationFromTwoPoints([LineX1, LineY1], [LineX2, LineY2], K, G, B):-

        % Наклонная прямая
        LineX1 <> LineX2, LineY1 <> LineY2,
        G = -1,
        K = (LineY2 - LineY1) / (LineX2 - LineX1),
        B = LineY1 - K * LineX1;

        % Вертикальная прямая
        LineX1 = LineX2, LineY1 <> LineY2,
        G = 0,
        K = -1,
        B = LineX1;

        % Горизонтальная прямая
        LineX1 <> LineX2, LineY1 = LineY2,
        G = -1,
        K = 0,
        B = LineY1.

% Поиск точек перечечения окружности со сторонами ромба
 getIntersections([Point1, Point2, Point3, Point4], CircleCentre, Radius, Answer):-
        getSegmentIntersection(Point1, Point2, CircleCentre, Radius, [A1, A2]),
        getSegmentIntersection(Point2, Point3, CircleCentre, Radius, [A3, A4]),
        getSegmentIntersection(Point3, Point4, CircleCentre, Radius, [A5, A6]),
        getSegmentIntersection(Point4, Point1, CircleCentre, Radius, [A7, A8]),

        set([A1, A2, A3, A4, A5, A6, A7, A8], Answer).

% Точка лежит на отрезке
 onSegment([Point1X, Point1Y], [Point2X, Point2Y], [PointX, PointY]):-
        min(Point1X, Point2X, MinX),
        max(Point1X, Point2X, MaxX),
        equalOrHigher(MinX, PointX),
        equalOrHigher(PointX, MaxX),

        min(Point1Y, Point2Y, MinY),
```

```
        max(Point1Y, Point2Y, MaxY),
        equalOrHigher(MinY, PointY),
        equalOrHigher(PointY, MaxY).

% Вернуть значение точки, если она лежит на отрезке
 getOnSegmentPoint(SegmentPoint1, SegmentPoint2, Point, Answer):-
        onSegment(SegmentPoint1, SegmentPoint2, Point),
        Answer = [Point];
        Answer = [[]].

% Поиск точек перечения отрезков и окружности
 getSegmentIntersection(SegmentPoint1, SegmentPoint2, CircleCentre, Radius, Answer):-
        lineEquationFromTwoPoints(SegmentPoint1, SegmentPoint2, K, G, B),
        getLineIntersection(CircleCentre, Radius, K, G, B, Point1, Point2),
        getOnSegmentPoint(SegmentPoint1, SegmentPoint2, Point1, AnswerPoint1),

        getOnSegmentPoint(SegmentPoint1, SegmentPoint2, Point2, AnswerPoint2),
        append(AnswerPoint1, AnswerPoint2, Answer);
        Answer = [[], []], true.

% Поиск точек перечения прямой и окружности
 getLineIntersection([CentreX, CentreY], Radius, K, G, B, [X1, Y1], [X2, Y2]):-

        %-----------------------------------------------
        % Решение квадратного уравнения относительно X
        G <> 0,
        Aq = ( K * K ) / ( G * G ) + 1,
        Bq = (2 * K * (B + CentreY * G) / (G * G)) - 2 * CentreX,
        Cq = (B / G + CentreY) * (B / G + CentreY) + (CentreX * CentreX) - Radius * Radius,
        qEquation(Aq, Bq, Cq, X1, X2),
        lineEquation(K, B, X1, Y1),
        lineEquation(K, B, X2, Y2);

        %-----------------------------------------------
        % Решение  квадратного уравнения относительно Y
        G = 0,
        Aq = 1,
        Bq = -2 * CentreY,
        Cq = (B - CentreX) * (B - CentreX) - Radius * Radius + CentreY * CentreY,
        qEquation(Aq, Bq, Cq, Y1, Y2),
        X1 = B, X2 = B.


% Общий вид уравнения:
%       R^2 = (X - x_c) ^ 2 + (-(kX + b)/g - y_c) ^ 2

% При G(g) <> 0:
%       R^2 = (X - x_c) ^ 2 + (-(kX + b)/g - y_c) ^ 2
%       X ^ 2 - 2x_c * X + x_c ^ 2 + (k * X + b + y_c * g) ^ 2 / (g ^ 2) - R ^ 2 = 0
%       X ^ 2 - 2x_c * X + x_c ^ 2 + ((k * X)  ^ 2 + (2k * (b + y_c * g) * X) + (b - y_c * g) ^ 2) /
(g ^ 2) - R ^ 2 = 0
```

```
%       X ^ 2 - 2x_c * X + x_c ^ 2 + (k / g) ^ 2 * X ^ 2 + ((2k * (b + y_c * g) * X ) + (b - y_c * g)
^ 2) / (g ^ 2) - R ^ 2 = 0
%       [(k / g) ^ 2  + 1 ] * X ^ 2 - [ 2x_c ] * X + x_c ^ 2 + (2k * (b + y_c * g) / (g ^ 2)) * X + (b
- y_c * g) ^ 2 / (g ^ 2) - R ^ 2 = 0
%       ---------------------------------------------------------------------------------------------
%       [(k / g) ^ 2  + 1 ] * X ^ 2 + [ 2k * (b + y_c * g) / (g ^ 2) - 2x_c ] * X + [ (b / g + y_c) ^ 2
+ x_c ^ 2 - R ^ 2 ] = 0

% При G(g) = 0:
%       R^2 = (B - x_c) ^ 2 + (Y - y_c) ^ 2
%       R^2 = (B - x_c) ^ 2 + Y ^ 2 - (2y_c) * Y + y_c ^ 2
%      ------------------------------------------------------------
%       Y^2 - [ 2y_c ] *  Y  + [ y_c ^ 2 + (B - x_c) ^ 2 - R ^ 2 ] = 0
```

Модуль "MATH.PRO":

```
% ------------------ Математические функции ------------------

PREDICATES

 normalErrorRate(real, real)
 equal(real, real)
 equal(real, real, real, real)
 equalOrHigher(real, real)
 max(real, real, real)
 min(real, real, real)

 qEquationRoot(real, real, real, real)
 qEquationTwoRoots(real, real, real, real, real)
 qEquationOneRoot(real, real, real, real)
 qEquationSolver(real, real, real, real, real, real)
 qEquation(real, real, real, real, real)

CLAUSES

% ---------------------- Основные функции ----------------------

% Проверка погрешности вычислений
 normalErrorRate(Value1, Value2):-
        abs(Value1 - Value2) < currency.

% Сравнение чисел с заданной точностью
 equal(Value1, Value2):-
        normalErrorRate(Value1, Value2).

 equal(Value1, Value2, Value3, Value4):-
        equal(Value1, Value2),
        equal(Value3, Value4),
        equal(Value1, Value3).

% Оператор ">="
 equalOrHigher(Value1, Value2):-
```

```
        equal(Value1, Value2);
        Value1 < Value2.

% Нахождение максимума из двух чисел
 max(Value1, Value2, Result):-
        Value1 > Value2, Result = Value1, !;
        Result = Value2.

% Нахождение минимума из двух чисел
 min(Value1, Value2, Result):-
        Value1 < Value2, Result = Value1, !;
        Result = Value2.

% ---------- Решение квадратного уравнения ----------

% Нахождение корня квадратного уравнения
 qEquationRoot(DiskrSqrt, A, B, Root):-
        Root = (DiskrSqrt - B) / (2 * A).

% "D > 0"
 qEquationTwoRoots(D, A, B, Root1, Root2):-
        DSqrt = sqrt(D),
        NegDSqrt = -DSqrt,
        qEquationRoot(NegDSqrt, A, B, Root1),
        qEquationRoot(DSqrt, A, B, Root2).

% "D = 0"
 qEquationOneRoot(D, A, B, Root):-
        DSqrt = sqrt(D),
        qEquationRoot(DSqrt, A, B, Root).

% Нахождение решения квадратного уравнения
 qEquationSolver(D, A, B, C, Root1, Root2):-
        D > 0, qEquationTwoRoots(D, A, B, Root1, Root2);
        D = 0, qEquationOneRoot(D, A, B, Root1), Root2 = Root1.

% Квадратное уравнение с заданными коэффициентами
 qEquation(A, B, C, Root1, Root2):-
        Diskriminant = B * B - 4 * A * C,
        qEquationSolver(Diskriminant, A, B, C, Root1, Root2).
```

Модуль "STRUCTS.PRO":

```
% ---------- Структуры данных ----------

PREDICATES
 member(realList, realList2D)
 set(realList2D, realList2D)

 emptyList(realList)
 emptyList2D(realList2D)
 append(realList2D, realList2D, realList2D)
```

```
CLAUSES

% ----------------------- Множества -----------------------
%               (с исключением пустых значений)

 member(X,[X|_]).
 member(X,[_|Tail]) :- member(X, Tail).

 set([],[]).
 set([Head|Tail], [Head|Out]) :-
          not(member(Head,Tail)),
          not(emptyList(Head)),
          set(Tail, Out).

 set([Head|Tail], Out) :-
          member(Head, Tail),
          set(Tail, Out);
          emptyList(Head),
          set(Tail, Out).

% ------------------------ Списки -------------------------

 emptyList([]).

 emptyList2D([]).

 append([], List, List).
 append([Head|Tail], List, [Head|Result]) :-
          append(Tail, List, Result).
```

Модуль "IO.PRO":

```
% ----------------------- Ввод/вывод данных -----------------------
%               (включена проверка корректности ввода)

PREDICATES
 menu(integer)
 inputData(realList2D, realList, real, integer)
 inputFilePath(string, string)
 inputRhombus(realList2D)
 inputPoint(realList, string)
 inputCircle(realList, real)

 correctMenuOption(integer)
 correctInput(realList2D, real, realList2D)
 correctRhombus(realList2D, realList2D)
 correctCircle(real)

 outputData(realList2D, realList, real, integer)
 outputPoints(realList2D)
 outputAnswer(realList2D, integer)
```

```
 outputRhombus(reallist2D)
 outputCircle(realList, real)

CLAUSES

% -------------------- Ввод данных -----------------------

% Стартовое меню
 menu(Option):-
        nl, write("===== PROGRAM STARTED ====="), nl,
        write("1. I/O console"), nl,
        write("2. I/O file"), nl,
        readint(Option),
        correctMenuOption(Option), !;
        write("[ ERROR ] Incorrect input"), nl, fail.

% Ввод имени файла в системе
 inputFilePath(FilePath, DefaultFilePath):-
        write("Dafault path: ", DefaultFilePath), nl,
        write("Change? [n/any_key]"), nl,
        readchar(Action),
        Action = 'n', FilePath = DefaultFilePath, !;
        nl, write("New absolute filepath: "), readln(FilePath).

% Ввод данных из файла
 inputData(Rhombus, CircleCentre, Radius, 2) :-
        inputFilePath(FilePath, "input.txt"),
        existfile(FilePath),
        openread(stream, FilePath),
        readdevice(stream),
        readterm(realList2D, RhombusCoordinates),
        readterm(realList, CircleCentre),
        readreal(Radius),
        closefile(stream),
        readdevice(keyboard),
        correctInput(RhombusCoordinates, Radius, Rhombus).

% Ввод данных с консоли
 inputData(Rhombus, CircleCentre, Radius, 1) :-
        inputCircle(CircleCentre, Radius),
        correctCircle(Radius),

        inputRhombus(RhombusCoordinates),
        correctRhombus(RhombusCoordinates, Rhombus).

% Ввод параметров окружности
 inputCircle(CentrePoint, Radius):-
        write("-- ENTER CIRCLE COORDINATES --"), nl,
        inputPoint(CentrePoint, "Centre"),

        write("[ Radius size ]:"), nl,
        write("   Radius = "),
```

```
        readreal(Radius).

% Ввод параметров ромба
 inputRhombus([Point1, Point2, Point3, Point4]):-
        write("-- ENTER RHOMBUS COORDINATES --"), nl,
        inputPoint(Point1, "1"),
        inputPoint(Point2, "2"),
        inputPoint(Point3, "3"),
        inputPoint(Point4, "4").

% Ввод координат точки
 inputPoint([X, Y], PointDescription):-
        write("[ Point '", PointDescription, "' ]:"), nl,
        write("   X = "), readreal(X),
        write("   Y = "), readreal(Y).

% -------------------- Проверка ввода ----------------------

 correctMenuOption(Option):-
        Option = 1; Option = 2.

 correctInput(Rhombus, Radius, RhombusOrdered):-
        correctRhombus(Rhombus, RhombusOrdered),
        correctCircle(Radius).

% Обработка некорректного ввода окружности
correctCircle(Radius):-
        isCircle(Radius), !;
        write("[ ERROR ] It's not a circle"), nl, fail.

% Введённый четырехугольник - ромб
 correctRhombus([Point1, Point2, Point3, Point4], CorrectRhombus):-
        isRhombus([Point1, Point2, Point3, Point4]),
        CorrectRhombus = [Point1, Point2, Point3, Point4], !;

        isRhombus([Point1, Point3, Point2, Point4]),
        CorrectRhombus = [Point1, Point3, Point2, Point4], !;

        isRhombus([Point1, Point2, Point4, Point3]),
        CorrectRhombus = [Point1, Point2, Point4, Point3].

% Обработка некорректного ввода ромба
 correctRhombus(_, _):- write("[ ERROR ] It's not a rhombus"), nl, fail.

% --------------------- Вывод данных -------------------------

% Вывод данных в файл
 outputData(RhombusCoordinates, CircleCentre, Radius, 2) :-
        nl, write("Writing input data to the file..."), nl,
        inputFilePath(FilePath, "output.txt"),
        openwrite(stream, FilePath),
        writedevice(stream),
```

```
        outputData(RhombusCoordinates, CircleCentre, Radius, 1),

        closefile(stream),
        writedevice(screen),
        write("Input data was written successfully"), nl;
        write("[ ERROR ] Can't write input data to the file"), nl, fail.

outputData(RhombusCoordinates, CircleCentre, Radius, 1) :-
        outputRhombus(RhombusCoordinates), nl,
        outputCircle(CircleCentre, Radius), nl.

outputPoints([]).
outputPoints([Head|Tail]):- write("   ", Head), nl, outputPoints(Tail).

outputAnswer(Answer, 2):-
        nl, write("Writing answer to the file..."), nl
        inputFilePath(FilePath, "output.txt"),
        openwrite(stream, FilePath),
        writedevice(stream),
        outputAnswer(Answer, 1),
        closefile(stream),
        writedevice(screen),
        write("Solution was written successfully"), nl, !;
        write("[ ERROR ] Can't write solution to the file"), nl, fail.

outputAnswer(Answer, 1):-
        emptyList2D(Answer), write("No solution has been found"), nl, !;
        write("Intersection points:"), nl,
        outputPoints(Answer).

outputRhombus(Rhombus):-
        write("Rhombus:"), nl,
        outputPoints(Rhombus).

outputCircle(CircleCentre, Radius):-
        write("Circle:"), nl,
        write("   Centre = ", CircleCentre), nl,
        write("   Radius = ", Radius), nl.
```