Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное учреждение высшего образования

ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ СИСТЕМ УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ (ТУСУР)

Кафедра автоматизированных систем управления (АСУ)

ЗАДАЧИ НА ГЕНЕРАЦИЮ НЕСКОЛЬКИХ ВАРИАНТОВ РЕШЕНИЙ

Отчет по лабораторной работе №5

По дисциплине

«Функциональное и логическое программирование»

Студент гр. 431-3

\_\_\_\_\_\_\_\_\_\_\_\_\_ Е.П. Бекиш

(подпись)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(дата)

Руководитель:

Доцент кафедры АСУ

\_\_\_\_\_\_\_\_\_\_\_\_\_ С.М. Алфёров

(подпись)

Томск 2024

**Оглавление**

[1 Цель работы 3](#_Toc182745758)

[2 Задание на лабораторную работу 4](#_Toc182745759)

[3 Ход работы 5](#_Toc182745760)

[4 Тестирование 6](#_Toc182745761)

[5 Вывод 7](#_Toc182745762)

[Приложение А (обязательное) Листинг программы 8](#_Toc182745763)

# Цель работы

Получить навыки логического программирования для решения неоднозначных задач.

# Задание на лабораторную работу

Написать программу в соответствии с вариантом.

Задание по варианту №4:

Генератор позиций после 3-х шагов шахматного коня, без повторений (для исключения повторений можно использовать assert).

# Ход работы

В ходе выполнения работы был реализован механизм, который представляет собой модель движения трёх ходов коня на шахматной доске.

Первая строка устанавливает факт horse\_position/2 как динамический, что означает, что его можно изменять в процессе выполнения программы. Этот факт предназначен для хранения всех позиций коня на доске. Правило check\_board/2 проверяет, находится ли конь в пределах шахматной доски размером 8x8. Далее следуют восемь правил move\_horse/2, которые определяют различные варианты ходов коня по шахматной доске. Каждый из них проверяет, находится ли новая позиция (NewX, NewY) в пределах доски, не встречалась ли ранее, и если эти условия выполнены — добавляет в базу фактов с помощью assert и выводит информацию о ходе. В свою очередь правило generate\_3movements\_horse/2 представляет собой последовательность из трех ходов коня, в которой используется форматированный вывод для отображения информации о текущем ходе. Строка horse\_position(SecondX, SecondY) и аналогичная ей horse\_position(ThirdX, ThirdY) заносят в соответствующие переменные данные из внутренней базы данных. И далее вызывается правило horse\_movement/2, которое пробует для каждой такой пары все варианты перемещений.

Листинг кода лабораторной работы представлен в приложении А.1.

# Тестирование

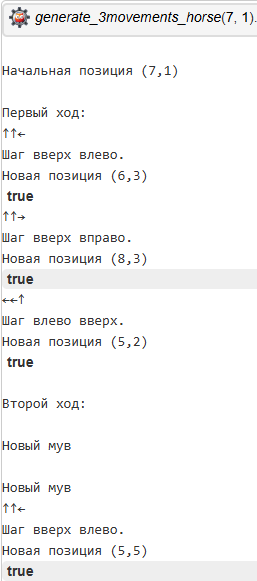
Таким образом, тестируя с начальной позицией (7,1), мы получаем результат, часть которого представлена на рисунке 4.1.

Рисунок 4.1 — Часть результата выполнения программы

# Вывод

В результате выполнения лабораторной работы я получил навыки логического программирования для решения неоднозначных задач.

# Приложение А (обязательное) Листинг программы

Листинг А.1 — Программа, реализующая генератор позиций после 3-х шагов шахматного коня

:-dynamic horse\_position/2.

% проверка на выход за пределы шахматной доски

check\_board(X, Y) :- X > 0, X < 9,

Y > 0, Y < 9.

% Шаг вверх влево ↑↑←

move\_horse(X, Y) :- NewX is X - 1, NewY is Y + 2,

check\_board(NewX, NewY),

not(horse\_position(NewX, NewY)),

format("↑↑←~nШаг вверх влево.~nНовая позиция (~w,~w)~n",

[NewX, NewY]),

assert(horse\_position(NewX, NewY)).

% Шаг вверх вправо ↑↑→

move\_horse(X, Y) :- NewX is X + 1, NewY is Y + 2,

check\_board(NewX, NewY),

not(horse\_position(NewX, NewY)),

format("↑↑→~nШаг вверх вправо.~nНовая позиция (~w,~w)~n",

[NewX, NewY]),

assert(horse\_position(NewX, NewY)).

% Шаг вправо вверх →→↑

move\_horse(X, Y) :- NewX is X + 2, NewY is Y + 1,

check\_board(NewX, NewY),

not(horse\_position(NewX, NewY)),

format("→→↑~nШаг вправо вверх.~nНовая позиция (~w,~w)~n",

[NewX, NewY]),

assert(horse\_position(NewX, NewY)).

% Шаг вправо вниз →→↓

move\_horse(X, Y) :- NewX is X + 2, NewY is Y - 1,

check\_board(NewX, NewY),

not(horse\_position(NewX, NewY)),

format("→→↓~nШаг вправо вниз.~nНовая позиция (~w,~w)~n",

[NewX, NewY]),

assert(horse\_position(NewX, NewY)).

% Шаг вниз вправо ↓↓→

move\_horse(X, Y) :- NewX is X + 1, NewY is Y - 2,

check\_board(NewX, NewY),

not(horse\_position(NewX, NewY)),

format("↓↓→~nШаг вниз вправо.~nНовая позиция (~w,~w)~n",

[NewX, NewY]),

assert(horse\_position(NewX, NewY)).

% Шаг вниз влево ↓↓←

move\_horse(X, Y) :- NewX is X - 1, NewY is Y - 2,

check\_board(NewX, NewY),

not(horse\_position(NewX, NewY)),

format("↓↓←~nШаг вниз влево.~nНовая позиция (~w,~w)~n",

[NewX, NewY]),

assert(horse\_position(NewX, NewY)).

% Шаг влево вниз ←←↓

move\_horse(X, Y) :- NewX is X - 2, NewY is Y - 1,

check\_board(NewX, NewY),

not(horse\_position(NewX, NewY)),

format("←←↓~nШаг влево вниз.~nНовая позиция (~w,~w)~n",

[NewX, NewY]),

assert(horse\_position(NewX, NewY)).

% Шаг влево вверх ←←↑

move\_horse(X, Y) :- NewX is X - 2, NewY is Y + 1,

check\_board(NewX, NewY),

not(horse\_position(NewX, NewY)),

format("←←↑~nШаг влево вверх.~nНовая позиция (~w,~w)~n",

[NewX, NewY]),

assert(horse\_position(NewX, NewY)).

horse\_movement(X, Y) :-

format("~nНовый мув~n"),

move\_horse(X, Y).

% ↑

% ←

% →

% ↓

generate\_3movements\_horse(FirstX, FirstY) :-

format("~nНачальная позиция (~w,~w)~n", [FirstX, FirstY]),

check\_board(FirstX, FirstY),

assert(horse\_position(FirstX, FirstY)),

% Расчёт первого хода

format("~nПервый ход:~n"),

move\_horse(FirstX, FirstY);

% Расчёт второго хода

format("~nВторой ход:~n"),

horse\_position(SecondX, SecondY),

horse\_movement(SecondX, SecondY);

% Расчёт третьего хода

format("~nТретий ход:~n"),

horse\_position(ThirdX, ThirdY),

horse\_movement(ThirdX, ThirdY).