Министерство образования Республики Беларусь

Учреждение образования

“Белорусский государственный университет

информатики и радиоэлектроники”

Факультет информационных технологий и управления

Кафедра интеллектуальных информационных технологий

**ЛАБОРАТОРНАЯ РАБОТА №2**

по дисциплине «Логические основы интеллектуальных систем»

**Вариант 4**

Выполнил Воложинец А. А.

ст. гр. 221703

Проверил Ивашенко В. П.

Минск 2024

# **Тема:** логическое программирование поиска решения задачи

# **Цель:** приобрести навыки логического программирования поиска решения задачи.

**Условие задания:**

Два берега реки. На одном из них полицейский с заключенным, мама с дочерьми и отец с сыновьями. Необходимо с помощью плота, вмещающего не более двух человек, иереправить всех персонажей на другой берег реки. Управлять плотом могут только полицейские и родители. Заключённого нельзя оставлять ни с одним из членов семьи. Папе не разрешается находиться с дочерьми без присутствия матери. Маме не разрешается находится с сыновьями без присутствия отца.

# **Дополнительные теоретические сведения**

Грамматика языка PROLOG.

<ПРОЛОГ-предложение> ::= <правило> | <факт> | <запрос>

<правило> ::= <заголовок> ‘:-’<тело>

<факт> ::= <заголовок> ‘.’

<запрос> ::= <тело>‘.’

<тело> ::= <цель> /’,’<цель>/’.’

<заголовок>::= <предикат>

<цель>::= <предикат> |<выражение>

<предикат>::= <имя>/ ‘(‘<терм> /’,’<терм>/ ‘)’/

<терм>::= <атом> |<предикат>|<список>

<атом>::= <переменная> |<число> |<строка>|<имя>

<список>::= <список с заголовком>| <простой список>

<список с заголовком >::= ‘[‘ <терм >/’,’<терм>/’|’ < терм>’]’

<простой список>::= ‘[‘ <терм >/’,’<терм>/’]’|‘['’]’

<выражение>::= <терм> /<оператор><терм>/

<оператор>::= ‘is’ | '=' | ‘==' | ’\=' | ’>=' | ’=<’ | ‘=\=' |

# **Описание программы и алгоритма:**

В рамках лабораторной работы стандартными средствами языка PROLOG был реализован алгоритм, позволяющий найти способ безопасного перемещения персонажей с одного берега на другой.

% Начало программы

run(Start, End) :-

go(Start, End).

go(Start, End) :- start(Start, End, [Start]).

% Решение задачи. Выбираются безопасные переправления, пока задача не будет решена.

start(State, End, Path) :-

move(State, NextState),

not(unsafe(NextState)),

not(member(NextState, Path)),

nl,

start(NextState, End, [NextState|Path]).

start(End, End, Path) :- print\_solution(Path).

print\_solution([State|Rest]) :-

print\_solution(Rest),

[NextState|\_] = Rest,

print\_move(NextState, State),

write(State), nl.

print\_solution([To, From]) :-

nl,

write(From), nl,

print\_move(From, To),

write(To), nl.

% Переправление полицейского

move([X, X, Father, Mother, Son1, Son2, Daughter1, Daughter2],

[Y, Y, Father, Mother, Son1, Son2, Daughter1, Daughter2]) :-

opposite(X, Y).

% Переправление отца

move([X, Convicted, X, Mother, Son1, Son2, Daughter1, Daughter2],

[Y, Convicted, Y, Mother, Son1, Son2, Daughter1, Daughter2]) :-

opposite(X, Y).

% Переправление матери

move([X, Convicted, Father, X, Son1, Son2, Daughter1, Daughter2],

[Y, Convicted, Father, Y, Son1, Son2, Daughter1, Daughter2]) :-

opposite(X, Y).

% Переправление полицейского с преступником

move([X, Convicted, Father, Mother, X, Son2, Daughter1, Daughter2],

[Y, Convicted, Father, Mother, Y, Son2, Daughter1, Daughter2]) :-

opposite(X, Y).

% Переправление полицейского с отцом

move([X, Convicted, Father, Mother, Son1, X, Daughter1, Daughter2],

[Y, Convicted, Father, Mother, Son1, Y, Daughter1, Daughter2]) :-

opposite(X, Y).

% Переправление полицейского с матерью

move([X, Convicted, Father, Mother, Son1, Son2, X, Daughter2],

[Y, Convicted, Father, Mother, Son1, Son2, Y, Daughter2]) :-

opposite(X, Y).

% Переправление полицейского с первым сыном

move([X, Convicted, Father, Mother, Son1, Son2, Daughter1, X],

[Y, Convicted, Father, Mother, Son1, Son2, Daughter1, Y]) :-

opposite(X, Y).

% Переправление полицейского с вторым сыном

move([X, Convicted, Father, Mother, Son1, Son2, Daughter1, Daughter2],

[Y, Convicted, Father, Mother, Son1, Son2, Daughter1, Daughter2]) :-

opposite(X, Y).

% Переправление полицейского с первой дочерью

move([P, Convicted, X, X, Son1, Son2, Daughter1, Daughter2],

[P, Convicted, Y, Y, Son1, Son2, Daughter1, Daughter2]) :-

opposite(X, Y).

% Переправление полицейского с второй дочерью

move([P, Convicted, X, Mother, X, Son2, Daughter1, Daughter2],

[P, Convicted, Y, Mother, Y, Son2, Daughter1, Daughter2]) :-

opposite(X, Y).

% Переправление отца с матерью

move([P, Convicted, X, Mother, Son1, X, Daughter1, Daughter2],

[P, Convicted, Y, Mother, Son1, Y, Daughter1, Daughter2]) :-

opposite(X, Y).

% Переправление отца с первым сыном

move([P, Convicted, X, Mother, Son1, Son2, Daughter1, Daughter2],

[P, Convicted, Y, Mother, Son1, Son2, Daughter1, Daughter2]) :-

opposite(X, Y).

% Переправление отца с вторым сыном

move([P, Convicted, Father, X, Son1, Son2, X, Daughter2],

[P, Convicted, Father, Y, Son1, Son2, Y, Daughter2]) :-

opposite(X, Y).

% Переправление матери с первой дочерью

move([P, Convicted, Father, X, Son1, Son2, Daughter1, X],

[P, Convicted, Father, Y, Son1, Son2, Daughter1, Y]) :-

opposite(X, Y).

% Переправление матери с второй дочерью

move([P, Convicted, Father, X, Son1, Son2, Daughter1, Daughter2],

[P, Convicted, Father, Y, Son1, Son2, Daughter1, Daughter2]) :-

opposite(X, Y).

% Замена местами аргументов

opposite(left, right).

opposite(right, left).

% Определение небезопасных переправлений

unsafe([H, X, X,\_,\_,\_,\_, \_]) :- opposite(H, X).

unsafe([H, X,\_, X,\_,\_,\_, \_]) :- opposite(H, X).

unsafe([H, X,\_,\_, X,\_,\_, \_]) :- opposite(H, X).

unsafe([H, X,\_,\_,\_, X,\_, \_]) :- opposite(H, X).

unsafe([H, X,\_,\_,\_,\_, X, \_]) :- opposite(H, X).

unsafe([H, X,\_,\_,\_,\_, \_, X]) :- opposite(H, X).

unsafe([\_,\_, H, X, X,\_,\_, \_]) :- opposite(H, X).

unsafe([\_,\_, H, X,\_, X,\_, \_]) :- opposite(H, X).

unsafe([\_,\_, X, H,\_,\_, X, \_]) :- opposite(H, X).

unsafe([\_,\_, X, H,\_,\_, \_, X]) :- opposite(H, X).

print\_move([From, From, Father, Mother, Son1, Son2, Daughter1, Daughter2],

[To, To, Father, Mother, Son1, Son2, Daughter1, Daughter2]) :-

opposite(From, To),

write("Полицейский с заключённым переправляются на "),

write\_side(To).

print\_move([From, Convicted, From, Mother, Son1, Son2, Daughter1, Daughter2],

[To, Convicted, To, Mother, Son1, Son2, Daughter1, Daughter2]) :-

opposite(From, To),

write("Полицейский с отцом переправляются на "),

write\_side(To).

print\_move([From, Convicted, Father, From, Son1, Son2, Daughter1, Daughter2],

[To, Convicted, Father, To, Son1, Son2, Daughter1, Daughter2]) :-

opposite(From, To),

write("Полицейский с мамой переправляются на "),

write\_side(To).

print\_move([From, Convicted, Father, Mother, From, Son2, Daughter1, Daughter2],

[To, Convicted, Father, Mother, To, Son2, Daughter1, Daughter2]) :-

opposite(From, To),

write("Полицейский с первым сыном переправляются на "),

write\_side(To).

print\_move([From, Convicted, Father, Mother, Son1, From, Daughter1, Daughter2],

[To, Convicted, Father, Mother, Son1, To, Daughter1, Daughter2]) :-

opposite(From, To),

write("Полицейский с вторым сыном переправляются на "),

write\_side(To).

print\_move([From, Convicted, Father, Mother, Son1, Son2, From, Daughter2],

[To, Convicted, Father, Mother, Son1, Son2, To, Daughter2]) :-

opposite(From, To),

write("Полицейский с первой дочерью переправляются на "),

write\_side(To).

print\_move([Policeman, Convicted, Father, From, Son1, Son2, From, Daughter2],

[Policeman, Convicted, Father, To, Son1, Son2, To, Daughter2]) :-

opposite(From, To),

write("Мама с первой дочерью переправляются на "),

write\_side(To).

print\_move([Policeman, Convicted, Father, From, Son1, Son2, Daughter1, From],

[Policeman, Convicted, Father, To, Son1, Son2, Daughter1, To]) :-

opposite(From, To),

write("Мама с второй дочерью переправляются на "),

write\_side(To).

print\_move([Policeman, Convicted, From, From, Son1, Son2, Daughter1, Daughter2],

[Policeman, Convicted, To, To, Son1, Son2, Daughter1, Daughter2]) :-

opposite(From, To),

write("Мама с отцом переправляются на "),

write\_side(To).

print\_move([Policeman, Convicted, From, Mother, From, Son2, Daughter1, Daughter2],

[Policeman, Convicted, From, Mother, To, Son2, Daughter1, Daughter2]) :-

opposite(From, To),

write("Отец с первым сыном переправляются на "),

write\_side(To).

print\_move([Policeman, Convicted, From, Mother, Son1, From, Daughter1, Daughter2],

[Policeman, Convicted, To, Mother, Son1, To, Daughter1, Daughter2]) :-

opposite(From, To),

write("Отец с вторым сыном переправляются на "),

write\_side(To).

print\_move([From, Convicted, Father, Mother, Son1, Son2, Daughter1, Daughter2],

[To, Convicted, Father, Mother, Son1, Son2, Daughter1, Daughter2]) :-

opposite(From, To),

write("Полицейский переправляется на "),

write\_side(To).

print\_move([Policeman, Convicted, Father, Mother, Son1, Son2, Daughter1, Daughter2],

[Policeman, Convicted, To, Mother, Son1, Son2, Daughter1, Daughter2]) :-

opposite(From, To),

write("Отец переправляется на "),

write\_side(To).

print\_move([Policeman, Convicted, Father, From, Son1, Son2, Daughter1, Daughter2],

[Policeman, Convicted, Father, To, Son1, Son2, Daughter1, Daughter2]) :-

opposite(From, To),

write("Мама переправляется на "),

write\_side(To).

%dummy print

print\_move([\_, \_, \_, \_, \_, \_, \_, \_], [\_, \_, \_, \_, \_, \_, \_, \_]).

write\_side(left) :-

write("начальный берег."), nl.

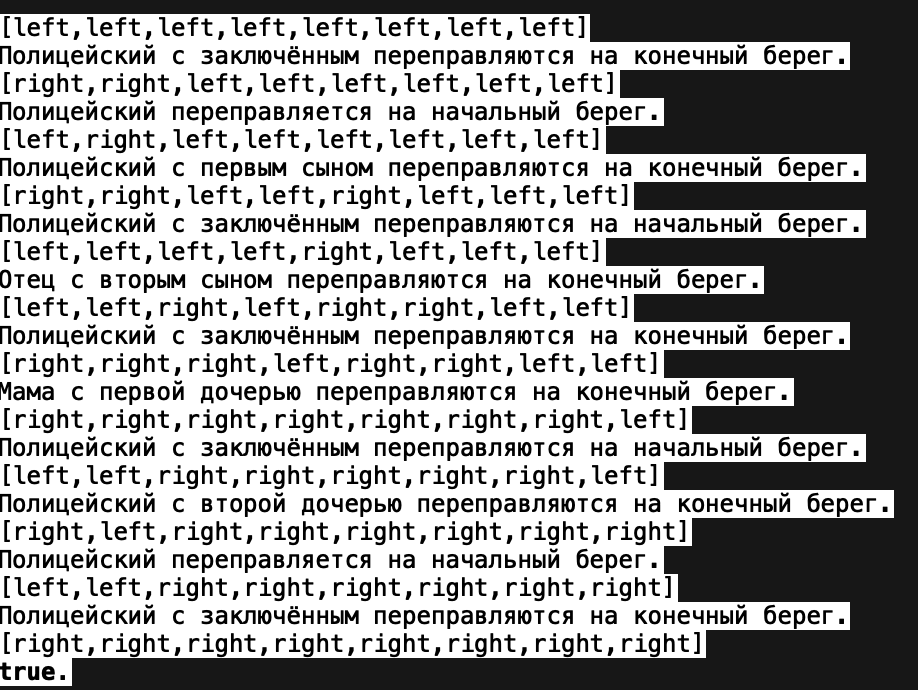
write\_side(right) :-

write("конечный берег."), nl.

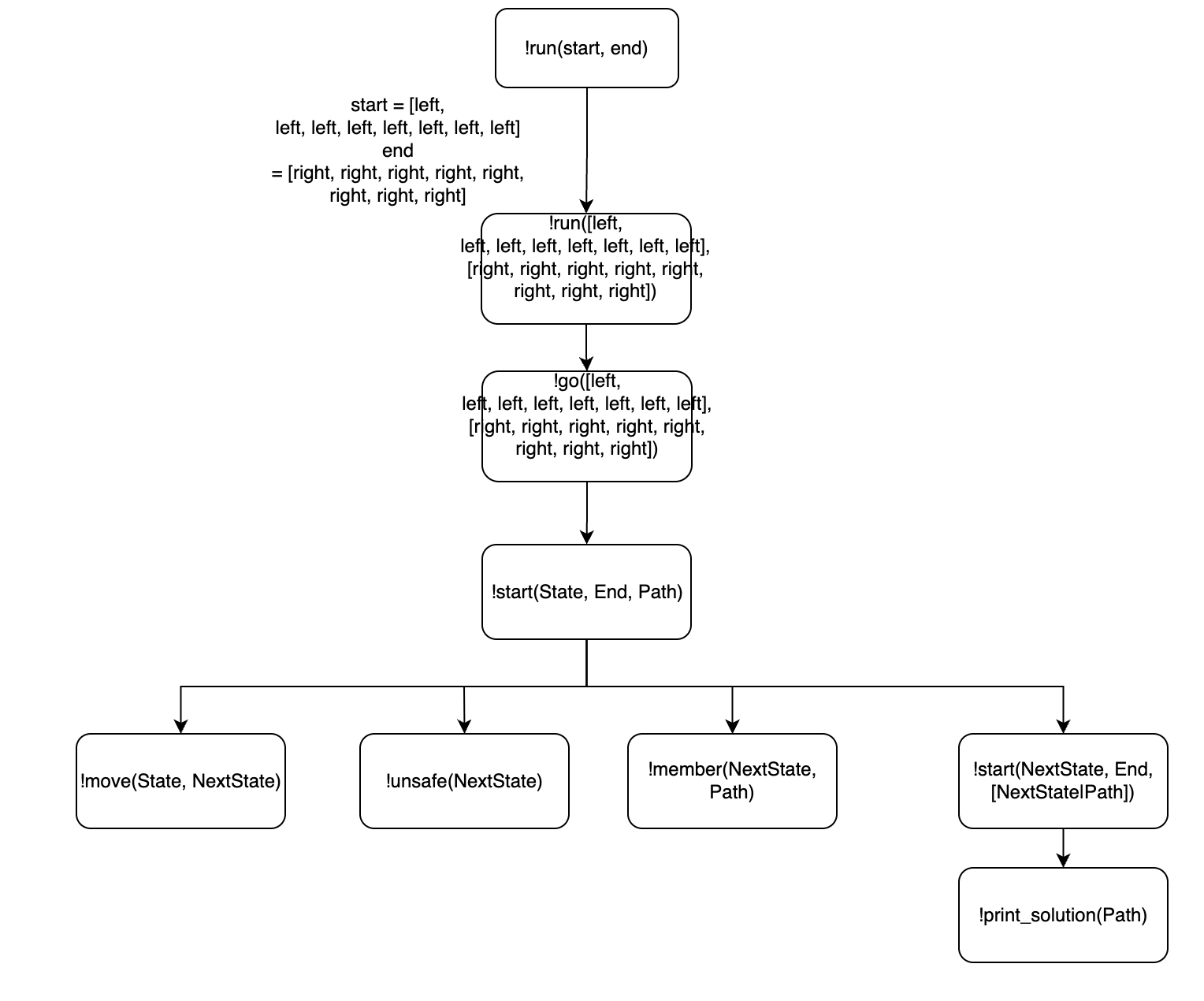
% Запуск программы с передачей начального и конечного состояния

:-run([left,left,left,left,left,left,left,left], [right,right,right,right,right,right,right,right]).

**Результат**

****

**Дерево вывода**



**Вывод**

В ходе выполнения лабораторной работы были приобретены навыки логического программирования поиска решения задачи; была разработана программа, решающая задачу о переправе полицейского, отца, сыновей, матери, дочерей.