Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ «Информатика и системы управления»

КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»

Лабораторная работа № 4
по дисциплине «Функциональные и логические языки программирования»

Тема Пролог. Лабиринт.

Студент Одинцов Е.В.

Группа ИУ7-53БВ

Преподаватели Строганов Ю.В.

Москва, 2024

Содержание

ВВЕДЕНИЕ

В рамках данной лабораторной работы было необходимо разработать программу на языке SWI-Prolog для решения задачи поиска выхода из лабиринта. Лабиринт содержит специальные клетки, такие как телепорты, ловушки и стены, и игрок может передвигаться как на соседние клетки, так и через одну клетку (прыжок).

Цель работы — изучить основы логического программирования, реализовать алгоритм поиска пути в лабиринте с учетом особенностей игрового пространства и специальных клеток.

1 Аналитическая часть

Поиск выхода из лабиринта можно свести к поиску путей на графе. Лабиринт можно представить как граф, где каждая клетка является вершиной, а возможные ходы игрока — рёбрами графа.

В данной задаче особые клетки (телепорты, ловушки и стены) изменяют поведение графа:

— Телепорт перемещает игрока на указанную позицию, что соответствует добавлению рёбер, соединяющих несмежные вершины.

— Ловушка может блокировать движение игрока через клетку, если количество шагов, сделанных до этого, кратно определённому числу.

— Стены запрещают движение через соответствующие клетки, что исключает такие клетки из графа.

Для поиска пути через такой лабиринт используются алгоритмы поиска по графам, например, поиск в глубину (DFS) или поиск в ширину (BFS). В данной работе для реализации выбрана логика поиска с использованием рекурсивных вызовов в SWI-Prolog, так как этот язык хорошо подходит для описания таких задач в терминах логических утверждений и отношений.

2 Технологическая часть

2.1 Описание программы

1) Лабиринт представлен в виде матрицы $M \times N$, где каждая ячейка может содержать одно из следующих значений:

— `p` — пустая клетка, по которой можно передвигаться.

— `w` — стена, через которую нельзя пройти.

— `t(TX, TY)` — телепорт, который перемещает игрока в указанную позицию `(TX, TY)`.

— `trap(K)` — ловушка, которая срабатывает каждые K шагов.

2) Игрок может перемещаться в четырёх направлениях: вверх, вниз, влево или вправо, а также совершать "прыжки"через одну клетку.

2.2 Пример кода

```
% Карта лабиринта: каждая ячейка описывается через map(X, Y, Содержимое).
map(1, 1, p).
map(1, 2, w).
map(1, 3, p).
map(1, 4, t(2, 2)).
map(2, 1, p).
map(2, 2, p).
map(2, 3, trap(3)).
map(2, 4, p).

% Позиция выхода
exit(2, 4).

% Перемещение игрока (обычное и прыжок через клетку)
move(X, Y, NX, Y) :- NX is X + 1.  % вправо
move(X, Y, NX, Y) :- NX is X - 1.  % влево
move(X, Y, X, NY) :- NY is Y + 1.  % вниз
move(X, Y, X, NY) :- NY is Y - 1.  % вверх
move(X, Y, NX, Y) :- NX is X + 2.  % прыжок вправо
move(X, Y, NX, Y) :- NX is X - 2.  % прыжок влево
move(X, Y, X, NY) :- NY is Y + 2.  % прыжок вниз
move(X, Y, X, NY) :- NY is Y - 2.  % прыжок вверх

% Проверка, можно ли идти на клетку (не является стеной)
```

```
can_move(X, Y) :- map(X, Y, p).
can_move(X, Y) :- map(X, Y, t(_, _)).  % Телепорт
can_move(X, Y) :- map(X, Y, trap(_)).  % Ловушка

% Поиск выхода
find_exit(X, Y, _, _) :-
    exit(X, Y),
    format('Выход найден на (~w, ~w)~n', [X, Y]).

find_exit(X, Y, Visited, Steps) :-
    \+ member((X, Y), Visited),
    can_move(X, Y),
    map(X, Y, t(TX, TY)),
    find_exit(TX, TY, [(X, Y) | Visited], Steps).

find_exit(X, Y, Visited, Steps) :-
    \+ member((X, Y), Visited),
    can_move(X, Y),
    move(X, Y, NX, NY),
    find_exit(NX, NY, [(X, Y) | Visited], Steps + 1).

% Начать поиск
start :-
    start(1, 1).

start(X, Y) :-
    find_exit(X, Y, [], 0).
```

ЗАКЛЮЧЕНИЕ

В ходе выполнения лабораторной работы была разработана программа на языке SWI-Prolog для поиска выхода из лабиринта, содержащего телепорты, ловушки и стены. Программа успешно реализует рекурсивный поиск пути с учетом всех особенностей лабиринта, а также использование телепортов и проверку на срабатывание ловушек.

Полученные результаты демонстрируют возможности языка SWI-Prolog в решении задач поиска путей, а также позволяют закрепить навыки программирования в логической парадигме.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. SWI-Prolog. Доступно на: https://www.swi-prolog.org/. [Дата обращения: сентябрь 2024].

2. Примеры использования библиотеки CLPFD. Доступно на: https://www.youtube.com/watch?v=sHo6-hk21L8. [Дата обращения: сентябрь 2024].