## Код программы

:- use\_module(library(http/thread\_httpd)).

:- use\_module(library(http/http\_dispatch)).

:- use\_module(library(http/http\_error)).

:- use\_module(library(http/http\_json)).

:- use\_module(library(http/http\_open)).

:- use\_module(library(http/json)).

:- use\_module(library(http/http\_client)).

:- dynamic starting\_pos/2, cell/3, field/4.

:- initialization server.

% Создание стартовой позиции

create\_pos(X, Y):-  retractall(starting\_pos(\_, \_)),

                    asserta(starting\_pos(X, Y)).

% Обнуление знаний о всех клетках (о всём поле)

reset\_cells():- retractall(cell(\_, \_, \_)).

/\* Создание поля из словаря \*/

create\_cell(X0, Y0, "player"):- assertz(cell(X0, Y0, empty)), create\_pos(X0, Y0).

create\_cell(X, Y, "empty"):-    assertz(cell(X, Y, empty)).

create\_cell(X, Y, "portal"):-   assertz(cell(X, Y, portal)).

create\_cell(X, Y, "block1"):-   assertz(cell(X, Y, block1)).

create\_cell(X, Y, "block2"):-   assertz(cell(X, Y, block2)).

create\_cell(X, Y, "block4"):-   assertz(cell(X, Y, block4)).

create\_cell(X, Y, "finish"):-   assertz(cell(X, Y, finish)).

create\_all\_cells([]):-!.

create\_all\_cells([H|T]):-create\_cell(H.x, H.y, H.cell), create\_all\_cells(T).

create\_pole(Dict):- reset\_cells(),

                    Pole = Dict.pole,

                    create\_all\_cells(Pole).

/\* Получаем состояние поля с игроком в координате X, Y, в ход N \*/

get\_blocks1(Blocks):-

    findall(json{x:X, y:Y, n:0}, cell(X, Y, block1), Blocks).

get\_blocks2(N, Blocks):-

    T is (2-N) mod 2,

    findall(json{x:X, y:Y, n:T}, cell(X, Y, block2), Blocks).

get\_blocks4(N, Blocks):-

    T is (4-N) mod 4,

    findall(json{x:X, y:Y, n:T}, cell(X, Y, block4), Blocks).

get\_finish(Finish):-

    cell(X, Y, finish),

    Finish = json{x:X, y:Y}.

get\_portals(Portals):-

    findall(json{x:X, y:Y}, cell(X, Y, portal), Portals).

get\_pole(Pole, X, Y, N):-

    Player = json{x:X, y:Y},

    get\_blocks1(Blocks1),

    get\_blocks2(N, Blocks2),

    get\_blocks4(N, Blocks4),

    append(Blocks1, Blocks2, Blocks12),

    append(Blocks12, Blocks4, Blocks),

    get\_finish(Finish),

    get\_portals(Portals),

    Pole = json{player:Player, walls:Blocks, portals:Portals, finish:Finish}.

%очистка базы знаний для алгоритма A\*

reset\_fields():-retractall(field(\_, \_, \_, \_)).

%подготовка очереди для алгоритма A\*

setup\_queue(Queue):-

    reset\_fields(), %очищаем базу знаний для алгоритма A\*

    starting\_pos(X0, Y0), %получаем начальные координаты игрока

    get\_pole(Pole0, X0, Y0, 0), %полуаем начальное состояние поля

    assert(field(X0, Y0, 0, [Pole0])), %добавляем первую информацию для алгоритма A\*

    Queue = [field(X0, Y0, 0, [Pole0])]. %И в очередь тоже

/\* Правила перемещения \*/

%если стоит вечная стена, не можем сюда пойти

move\_to\_cell(cell(\_, \_, block1), \_, \_, \_):-fail.

%если стоит стена2 и номер хода кратен 2, не можем сюда пойти

move\_to\_cell(cell(\_, \_, block2), N, \_, \_):-T is (N mod 2), T = 0, !, fail. %можно возвращать -1, -1 здесь, а не в check\_move

%иначе можем

move\_to\_cell(cell(X, Y, block2), \_, X, Y).

%если стоит стена4 и номер хода кратен 4, не можем сюда пойти

move\_to\_cell(cell(\_, \_, block4), N, \_, \_):-T is (N mod 4), T = 0, !, fail.

%иначе можем

move\_to\_cell(cell(X, Y, block4), \_, X, Y).

%если наступили в портал, то надо изменить X, Y на координаты другого портала

move\_to\_cell(cell(X, Y, portal), \_, I, J):-cell(I, J, portal), not(I=X), not(J=Y).

%если пустая клетка, то просто идём в нее

move\_to\_cell(cell(X, Y, empty), \_, X, Y).

%если финашная клетка, то просто идём в нее

move\_to\_cell(cell(X, Y, finish), \_, X, Y).

% проверяем можно ли пойти в X, Y

% и получаем новые координаты NX, NY (обычно равны X, Y если не портал)

check\_move(X, Y, N, NX, NY):-

    cell(X, Y, Cell),

    move\_to\_cell(cell(X, Y, Cell), N, NX, NY), !. %move\_to\_cell(cell(2, 1, portal), 2, D, E)

% если check\_move fails, то надо вернуть пустой список

check\_move(\_, \_, \_, -1, -1).

% если check\_move fails, то надо вернуть пустой список NewField = []

check\_field(-1, -1, \_, \_, []):-!.

% то же самое если уже есть знания в координате X, Y

% (уже найден такой же или более короткий путь в эту клетку)

check\_field(X, Y, \_, \_, []):-

    field(X, Y, \_, \_), !.

% если же мы можем попасть X, Y и еще не были здесь,

% то добавляем новые знания вида field(X, Y, N, NewPath)

% также возвращаем NewField = [field(X, Y, N, NewPath)]

check\_field(X, Y, N, PrevPath, NewField):-

    get\_pole(Pole, X, Y, N),

    append(PrevPath, [Pole], NewPath),

    assert(field(X, Y, N, NewPath)),

    NewField = [field(X, Y, N, NewPath)].

% пытаемся переместиться

% если успешно, то NewField = [field(X, Y, N, Path)]

% иначе Field = []

try\_to\_move(X, Y, N, PrevPath, NewField):-

    % нельзя fail

    check\_move(X, Y, N, NX, NY),

    check\_field(NX, NY, N, PrevPath, NewField).

% пытаемся переместиться

% если успешно, то Field = [field(X, Y, N, Path)]

% иначе Field = []

try\_to\_go\_right(field(X, Y, N, Path), Field):-

    NN is N+1,

    NX is X+1,

    try\_to\_move(NX, Y, NN, Path, Field).

try\_to\_go\_down(field(X, Y, N, Path), Field):-

    NN is N+1,

    NY is Y+1,

    try\_to\_move(X, NY, NN, Path, Field).

try\_to\_go\_left(field(X, Y, N, Path), Field):-

    NN is N+1,

    NX is X-1,

    try\_to\_move(NX, Y, NN, Path, Field).

try\_to\_go\_up(field(X, Y, N, Path), Field):-

    NN is N+1,

    NY is Y-1,

    try\_to\_move(X, NY, NN, Path, Field).

append5(L1, L2, L3, L4, L5, Res):-

    append(L1, L2, R12),

    append(R12, L3, R123),

    append(R123, L4, R1234),

    append(R1234, L5, Res).

% выходим из цикла если очередь пуста

while\_queue\_is\_not\_empty([]):-!.

% иначе обрабатываем первый элемент в очереди:

% получаем новые знание вида field(X, Y, N, Path) и добавляем их в очердь.

while\_queue\_is\_not\_empty([H|T]):-

    try\_to\_go\_right(H, Right),

    try\_to\_go\_down(H, Down),

    try\_to\_go\_left(H, Left),

    try\_to\_go\_up(H, Up),

    append5(T, Right, Down, Left, Up, NewQueue),

    while\_queue\_is\_not\_empty(NewQueue).

% Алгоритм A\*

azvezda():-

    %setup(), не нужно?

    setup\_queue(Queue), % подготавливаем очередь

    while\_queue\_is\_not\_empty(Queue). %запускаем цикл пока очередь не опустеет

% Получаем кратчайший путь

get\_min\_path(Path):-

    azvezda(),

    cell(X, Y, finish),

    field(X, Y, \_, Path),

    reset\_fields().

% ...

:- http\_handler(root(.), my\_func, []).

% мой функция handler

my\_func(Request):-

    reset\_cells(),

    http\_read\_json\_dict(Request, Dict),

    create\_pole(Dict),

    get\_min\_path(Path),

    reply\_json(json{pole:Path}).

% запуск сервера

server():-http\_server(http\_dispatch, [port(1234)]).